# ProMoVer

# Modular Verification of Temporal Safety Properties

Siavash Soleimanifard
KTH Royal Institute of Technology
Stockholm, Sweden

Joint work with:

Dilian Gurov
KTH Royal Institute of Technology
Stockholm, Sweden

Marieke Huisman
University of Twente
The Netherlands

SEFM 2011, Montevideo, Uruguay
November 2011

# Software Verification

# Software Verification

- Verification of realistic software

# Software Verification

- Verification of realistic software
  - algorithmic

# Software Verification

- Verification of realistic software
  - algorithmic
  - light weight

# Software Verification

- Verification of realistic software
  - algorithmic
  - light weight
  - modular (compositional)

# Software Verification

- Verification of realistic software
  - algorithmic
  - light weight
  - modular (compositional)
    - complex and large systems

# Software Verification

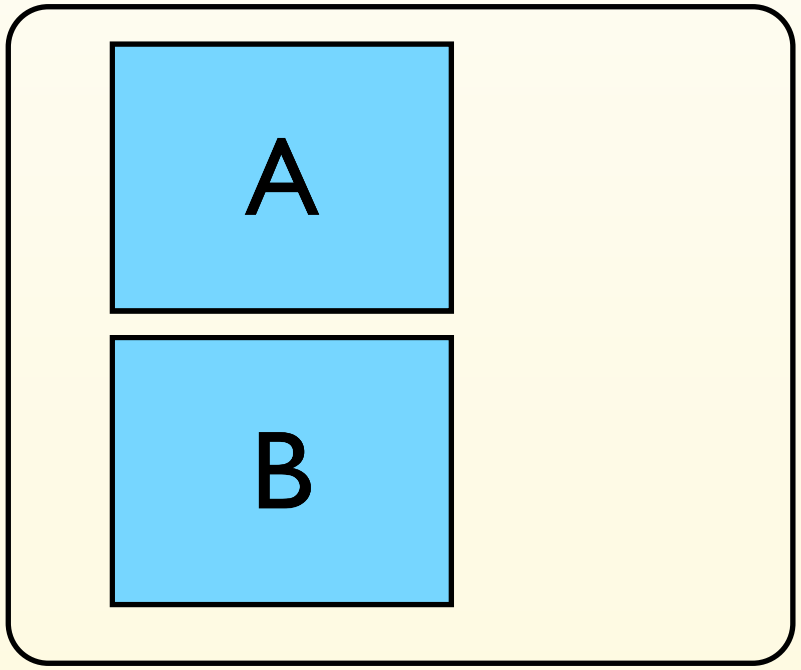
- Verification of realistic software
  - algorithmic
  - light weight
  - modular (compositional)
    - complex and large systems
    - facilitating the reuse of components

# Software Verification

- Verification of realistic software
  - algorithmic
  - light weight
  - modular (compositional)
    - complex and large systems
    - facilitating the reuse of components
    - support variability

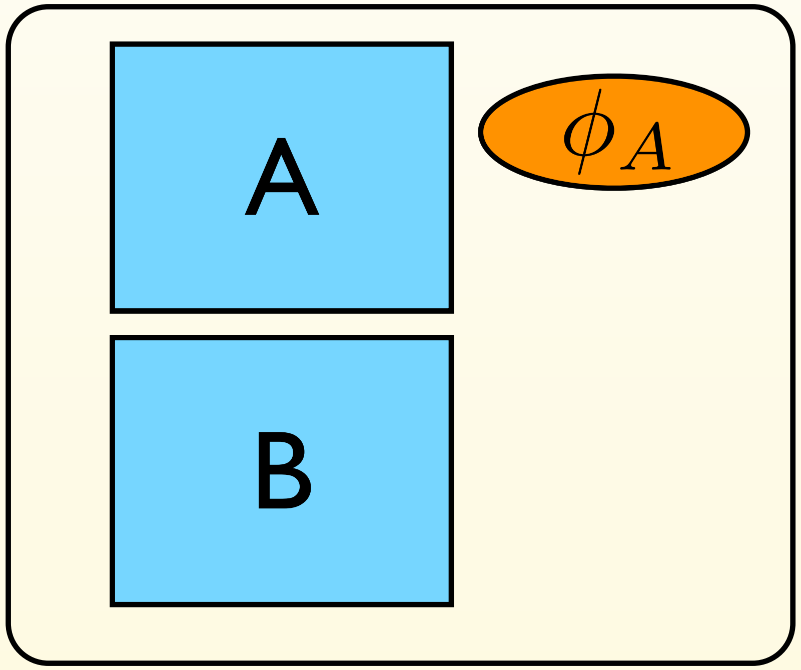
# Modular Verification

# Modular Verification



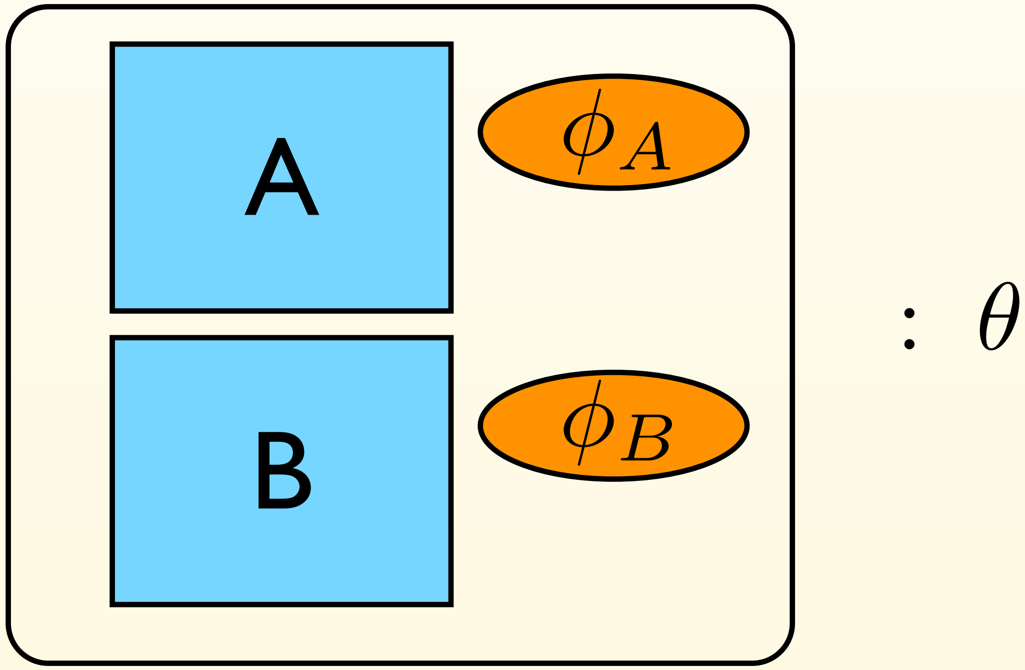
$: \theta$

# Modular Verification



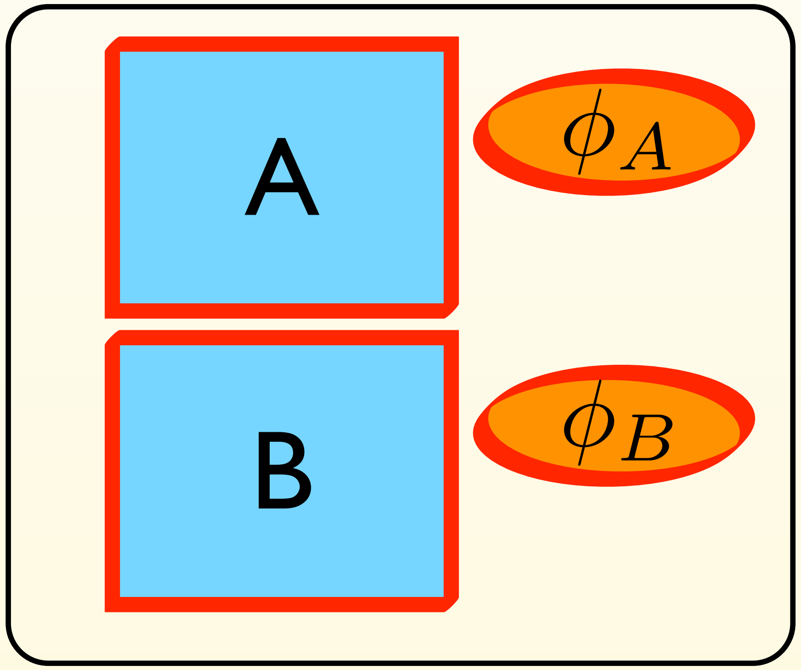
$: \ \theta$

# Modular Verification



# Modular Verification

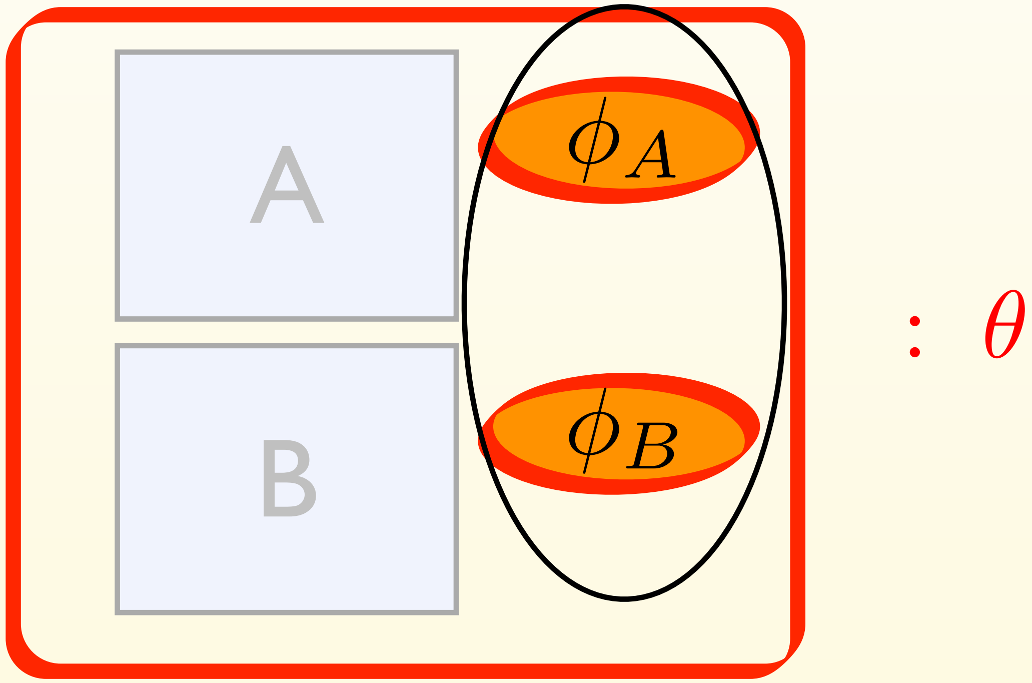
## Task I: Local Check



$: \;\theta$

# Modular Verification

## Task II: Global Check



# Outline of the Talk

- Our Approach
- Usage Example
- Verification based on Maximal Models
- Variability Scenarios
- ProMoVer
- Demo (ProMoVer web-interface)
- Conclusion & Future Work

# Our Approach

# Our Approach

- Algorithmic
  - Accepts an annotated Java program as input
  - Push-button tool support to verify the program (ProMoVer)
    - returns a positive answer or negative answer with a counter example

# Our Approach

- Algorithmic
  - Accepts an annotated Java program as input
  - Push-button tool support to verify the program (ProMoVer)
    - returns a positive answer or negative answer with a counter example
- Modular
  - modules are methods, e.g., Hoare logic

# Properties

# Properties

- The price of algorithmic approach is abstraction
  - We abstract away from all data
  - Flow graphs

# Properties

- The price of algorithmic approach is abstraction
  - We abstract away from all data
  - Flow graphs
- We consider temporal safety properties of the control flow
  - Legal sequence of method invocations

# Some Example Properties

# Some Example Properties

- Authorized access
  - method to change sensitive data is only called **within** authentication method

# Some Example Properties

- Authorized access
  - method to change sensitive data is only called **within** authentication method

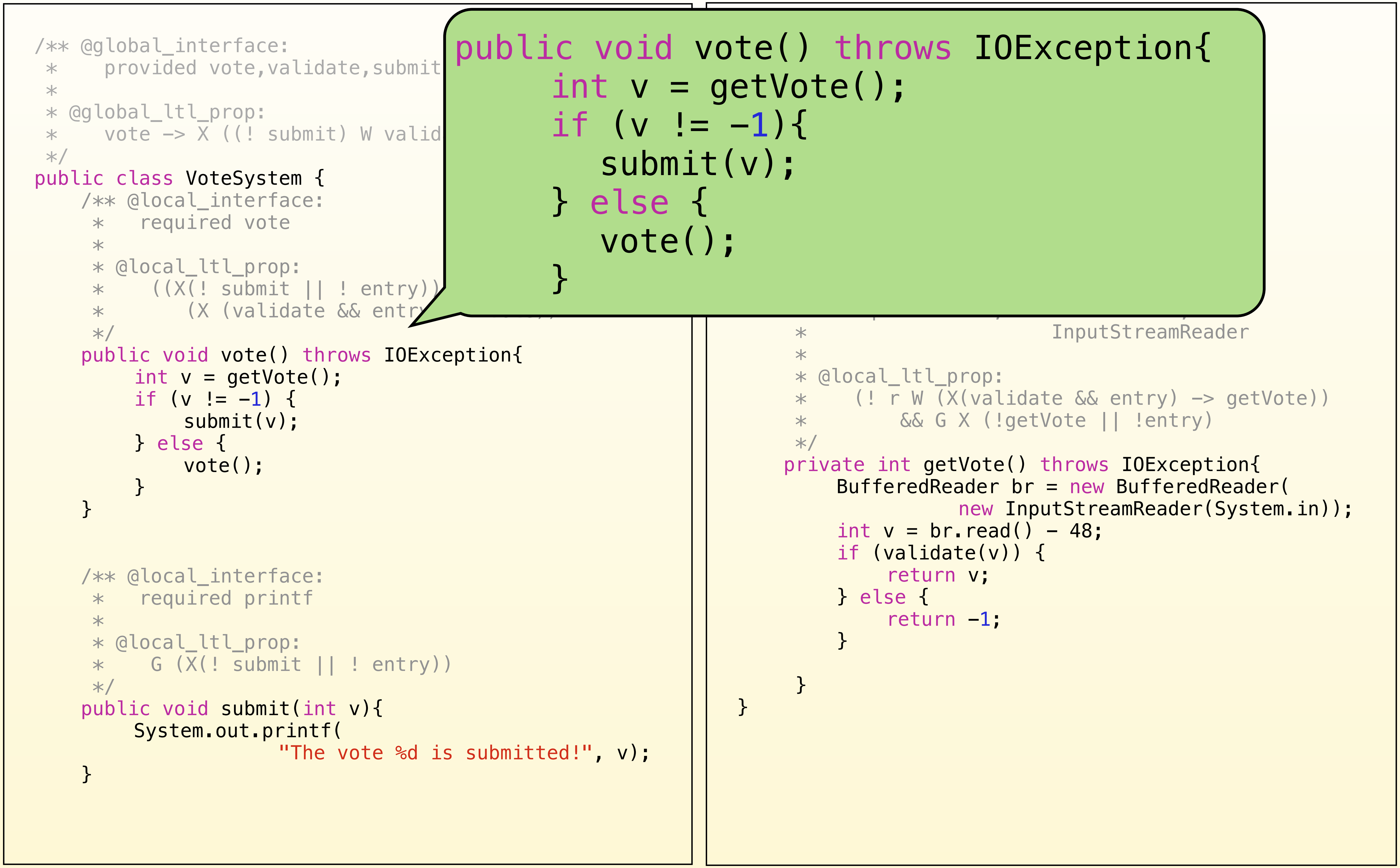
- Voting system
  - candidate selection has to be finished, **before** the vote can be confirmed

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate && entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                        InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

# Usage Example



```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W valid
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry))
     *       (X (validate && entr
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
public void vote() throws IOException{
        int v = getVote();
        if (v != -1){
           submit(v);
        } else {
           vote();
        }
```

```
     *                       InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                      new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate && entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
        System.out.printf(
                "The vote %d is submitted!", v);
    }
```

```
public void submit(int v){
     System.out.printf(
          "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }


    /** @local_interface:
     *   required read, BufferedReader,
     *                  InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *         && G X (!getVote || !entry)
                                                tion{
                                               dReader(
                                               (System.in));



            }

    }
}
```

# Usage Example

```java
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate && entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

```java
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                        InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

# Usage Example

```
private int getVote() throws IOException
{
     BufferedReader br = new
       BufferedReader(new
          InputStreamReader(System.in));
     int v = br.read() - 48;
     if (validate(v)) {
        return v;
     } else {
        return -1;
     }
    }
```

```
        } else {
            vote();
        }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
 @local_interface:
  required validate

@local_ltl_prop:
   G (X(! validate || ! entry))

lic boolean validate(int v){
 return ((1 <= v) && (v <= 5));



 @local_interface:
  required read, BufferedReader,
                 InputStreamReader

@local_ltl_prop:
   (! r W (X(validate && entry) -> getVote))
       && G X (!getVote || !entry)
*/
private int getVote() throws IOException{
     BufferedReader br = new BufferedReader(
                  new InputStreamReader(System.in));
     int v = br.read() - 48;
     if (validate(v)) {
         return v;
     } else {
         return -1;
     }

   }
}
```

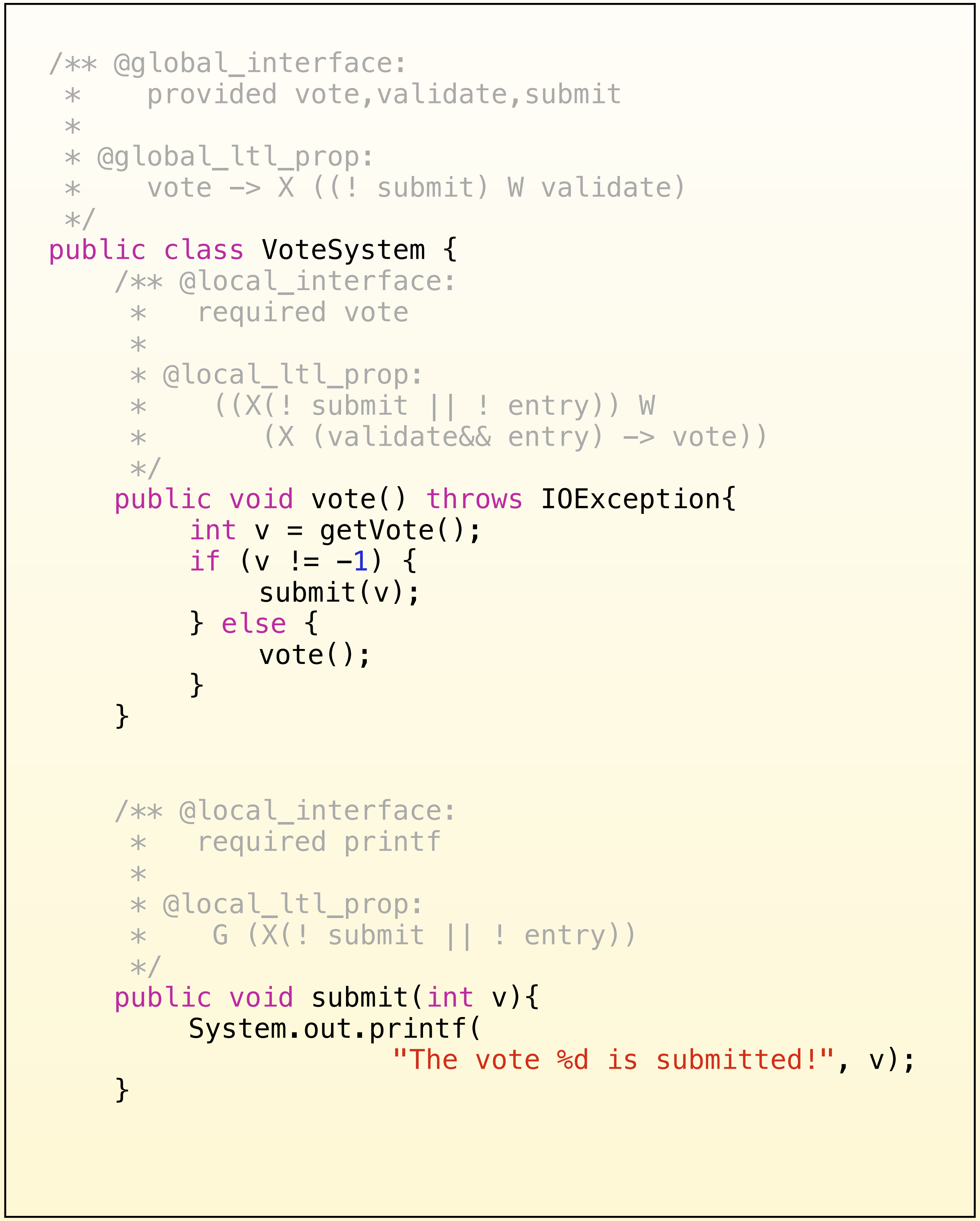
# mple

```
public boolean validate(int v){
     return ((1 <= v) &&(v <= 5));
  }
```

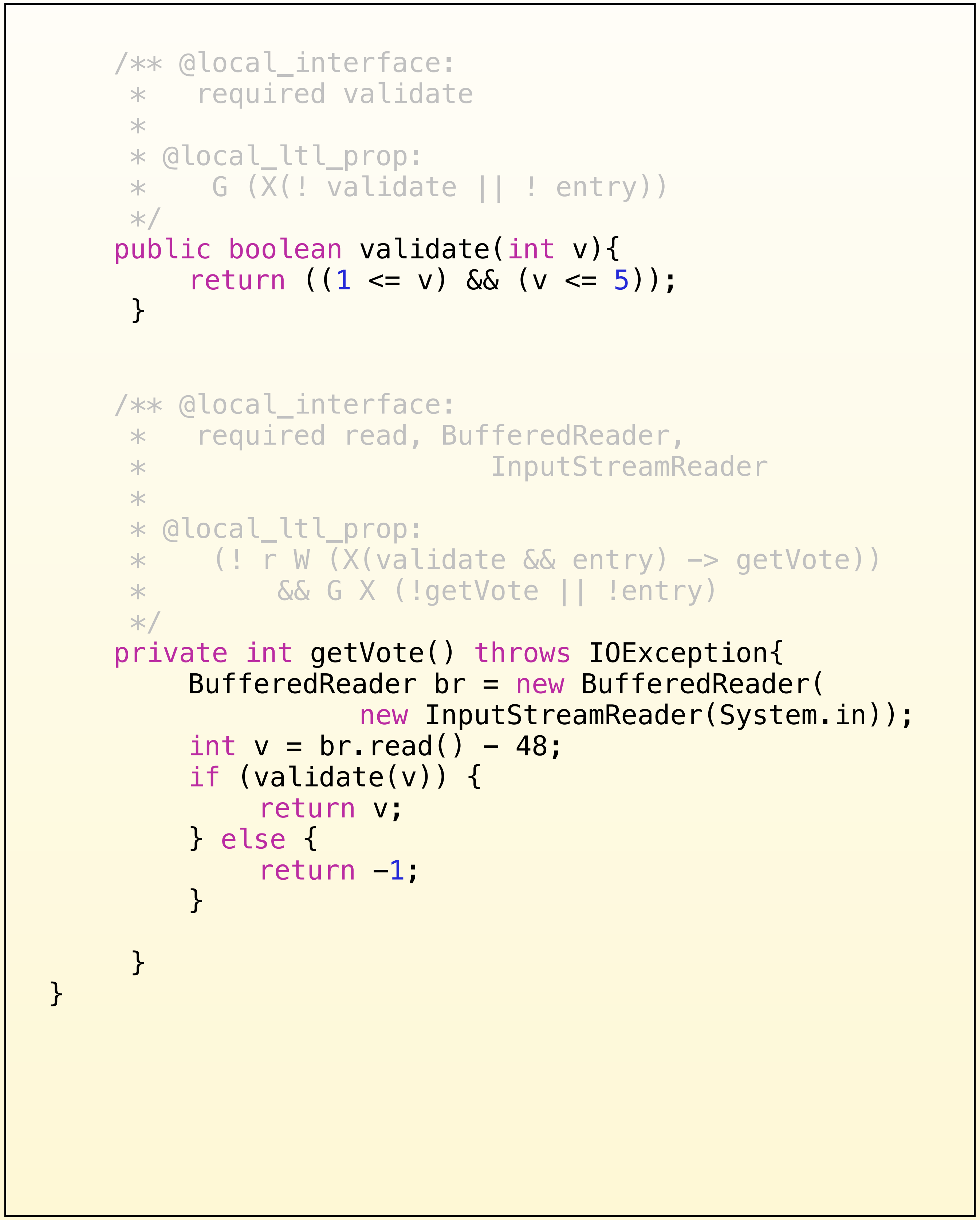
```
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
   /** @local_interface:
    *   required vote
    *
    * @local_ltl_prop:
    *    ((X(! submit || ! entry)) W
    *       (X (validate && entry) -> vote))
    */
   public void vote() throws IOException{
      int v = getVote();
      if (v != -1) {
         submit(v);
      } else {
         vote();
      }
   }


   /** @local_interface:
    *   required printf
    *
    * @local_ltl_prop:
    *    G (X(! submit || ! entry))
    */
   public void submit(int v){
      System.out.printf(
               "The vote %d is submitted!", v);
   }
```

```
      local_interface:
    *  equired validate
    *
    * @loc  l_ltl_prop:
    *  G  X(! validate || ! entry))
    */
   public boolean validate(int v){
      return ((1 <= v) && (v <= 5));
    }


   /** @local_interface:
    *   required read, BufferedReader,
    *                InputStreamReader
    *
    * @local_ltl_prop:
    *  (! r W (X(validate && entry) -> getVote))
    *     && G X (!getVote || !entry)
    */
   private int getVote() throws IOException{
      BufferedReader br = new BufferedReader(
             new InputStreamReader(System.in));
      int v = br.read() - 48;
      if (validate(v)) {
         return v;
      } else {
         return -1;
      }

    }
}
```

# Usage Example



```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```



```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                     InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
         BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
         int v = br.read() - 48;
         if (validate(v)) {
             return v;
         } else {
             return -1;
         }

     }
}
```

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *        (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
              submit(v);
         } else {
              vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

**No submit Until getVote**

```
     /** @local_interface:
      *   required validate
      *
      * @local_ltl_prop:
      *    G (X(! validate || ! entry))
      */
     public boolean validate(int v){
          return ((1 <= v) && (v <= 5));
      }


     /** @local_interface:
      *   required read, BufferedReader,
      *                      InputStreamReader
      *
      * @local_ltl_prop:
      *    (! r W (X(validate && entry) -> getVote))
      *        && G X (!getVote || !entry)
      */
     private int getVote() throws IOException{
          BufferedReader br = new BufferedReader(
                      new InputStreamReader(System.in));
          int v = br.read() - 48;
          if (validate(v)) {
               return v;
          } else {
               return -1;
          }

      }
}
```

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

**No submit Until getVote**

```
     *        (X (validated&& entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *
     *
     *
     *
     */
```

**No self call**

```
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *    required read, BufferedReader,
     *                        InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No submit Until getVote

```
     *          (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
              submit(v);
         } else {
              vote();
         }
    }


    /** @local_interface:
     *
     *
     *
     *
     */
```

No self call

```
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
      /** @local_interface:
       *    required validate
       *
       *
       *
       */
```

No self call

```
      public boolean validate(int v){
          return ((1 <= v) && (v <= 5));
       }


      /** @local_interface:
       *    required read, BufferedReader,
       *                   InputStreamReader
       *
       * @local_ltl_prop:
       *     (! r W (X(validate && entry) -> getVote))
       *         && G X (!getVote || !entry)
       */
      private int getVote() throws IOException{
          BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
          int v = br.read() - 48;
          if (validate(v)) {
              return v;
          } else {
              return -1;
          }

       }
  }
```

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No submit Until getVote

```
     *
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *
     *
     *
     *
     */
```

No self call

```
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

No self call

```
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
```

No return Until validate
& no self call

```
    private int getVote() throws IOException{
         BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
         int v = br.read() - 48;
         if (validate(v)) {
             return v;
         } else {
             return -1;
         }

     }
}
```

# Usage Example

If starting in **vote**,
no **submit** Until **validate**

No **submit** Until **getVote**

No self call

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *     required vote
     *
     *
     *
     *
     *          (X (validated entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *
     *
     *
     *
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

No self call

No **return** Until **validate**
& no self call

```
     /** @local_interface:
      *    required validate
      *
      *
      *
      */
     public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
      }


     /** @local_interface:
      *
      *
      *
      *
      *
      *
      */
     private int getVote() throws IOException{
         BufferedReader br = new BufferedReader(
                      new InputStreamReader(System.in));
         int v = br.read() - 48;
         if (validate(v)) {
             return v;
         } else {
             return -1;
         }

      }
 }
```

# Usage Example

**If starting in vote,**
**no submit Until validate**

No **submit** Until **getVote**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote

     *      (X (validated entry) -> vote))
     */
    public void vote() thr    s IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interfac
     *
     *
     *
     *
     */
    public void submit(i
        System.out.print
                  "The
    }
```

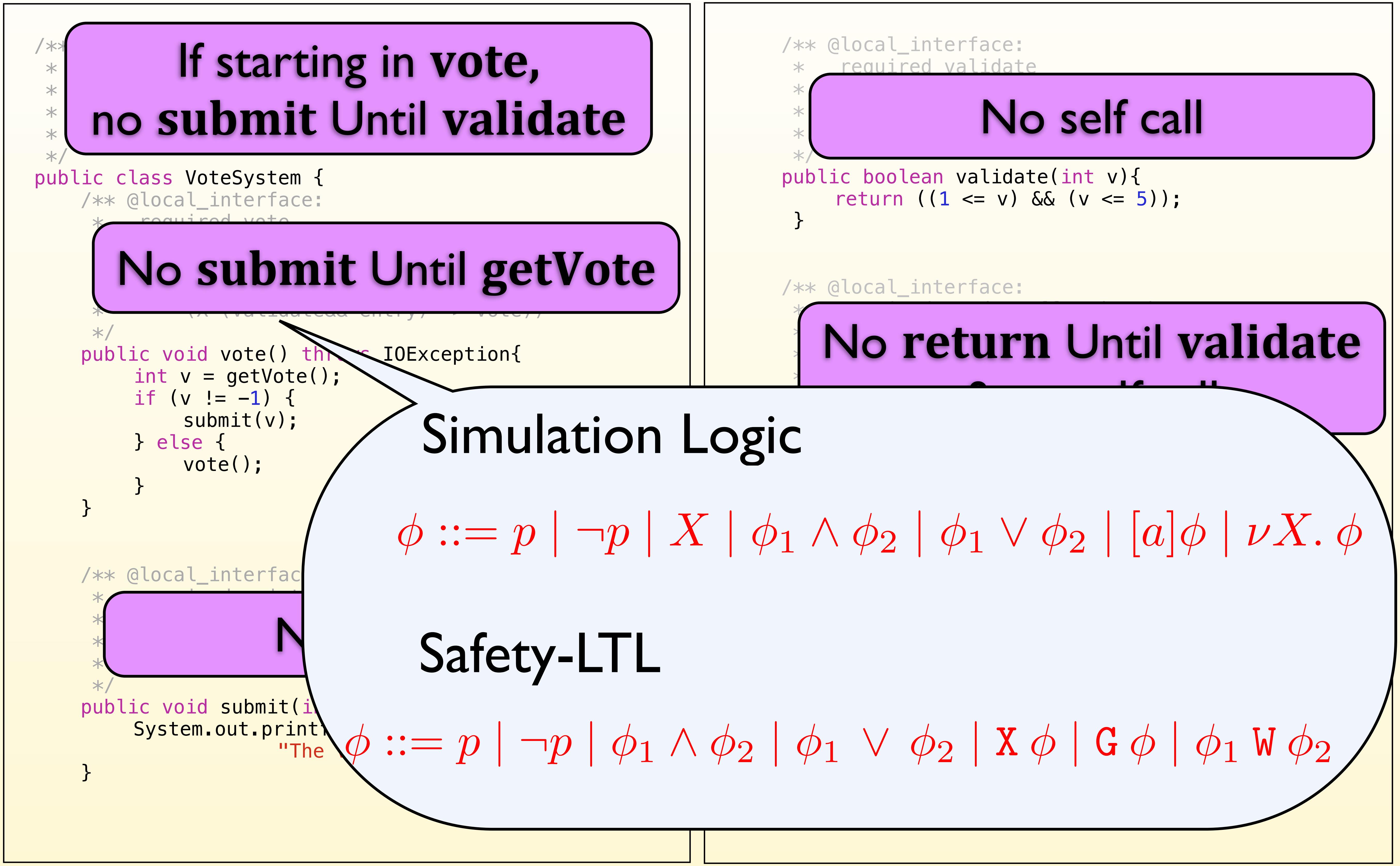
N

No self call

No **return** Until **validate**

```
/** @local_interface:
 *    required validate
 *
 *
 *
 */
public boolean validate(int v){
    return ((1 <= v) && (v <= 5));
}


/** @local_interface:
 *
 *
 *
```

# Usage Example



```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }

    /** @local_interfac
     *
     *
     *
     *
     */
    public void submit(i
        System.out.print
            "The
    }
```

```
/** @local_interface:
 *    required validate
 *
 *
 *
 */
public boolean validate(int v){
    return ((1 <= v) && (v <= 5));
 }

/** @local_interface:
 *
 *
 *
 */
```

Simulation Logic

$$\phi ::= p \mid \neg p \mid X \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid [a]\phi \mid \nu X.\, \phi$$

Safety-LTL

$$\phi ::= p \mid \neg p \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \mathtt{X}\, \phi \mid \mathtt{G}\, \phi \mid \phi_1\, \mathtt{W}\, \phi_2$$

# Usage Example

If starting in **vote**,
no **submit** Until **validate**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required_vote
```

No **submit** Until **getVote**

```
     *
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *
```

No self call

```
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                   "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
```

No self call

```
     *
     *
     *
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *
```

No **return** Until **validate**
& no self call

```
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

# Usage Example

```
/** @global_interface:
 *    provided vote,validate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                      InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
         BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
         int v = br.read() - 48;
         if (validate(v)) {
             return v;
         } else {
             return -1;
         }

     }
}
```

# Usage Example

YES

```
/** @global_interfac
 *    provided vote,va idate,submit
 *
 * @global_ltl_prop:
 *    vote -> X ((! submit) W validate)
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                       InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
 }
```

# Usage Example

**If starting in vote, never submit**

```
/*
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                 InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
         int v = br.read() - 48;
         if (validate(v)) {
             return v;
         } else {
             return -1;
         }

     }
}
```

# Usage Example

No C.E.

**If starting in vote, never submit**

```
/*
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *   required vote
     *
     * @local_ltl_prop:
     *    ((X(! submit || ! entry)) W
     *       (X (validate&& entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *   required printf
     *
     * @local_ltl_prop:
     *    G (X(! submit || ! entry))
     */
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *   required validate
     *
     * @local_ltl_prop:
     *    G (X(! validate || ! entry))
     */
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *   required read, BufferedReader,
     *                  InputStreamReader
     *
     * @local_ltl_prop:
     *    (! r W (X(validate && entry) -> getVote))
     *        && G X (!getVote || !entry)
     */
    private int getVote() throws IOException{
         BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
         int v = br.read() - 48;
         if (validate(v)) {
             return v;
         } else {
             return -1;
         }

     }
}
```

# Verification

If starting in **vote**,
no **submit** Until **validate**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required_vote
     *
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }
```

No **submit** Until **getVote**

```
    /** @local_interface:
     *    required_printf
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

No self call

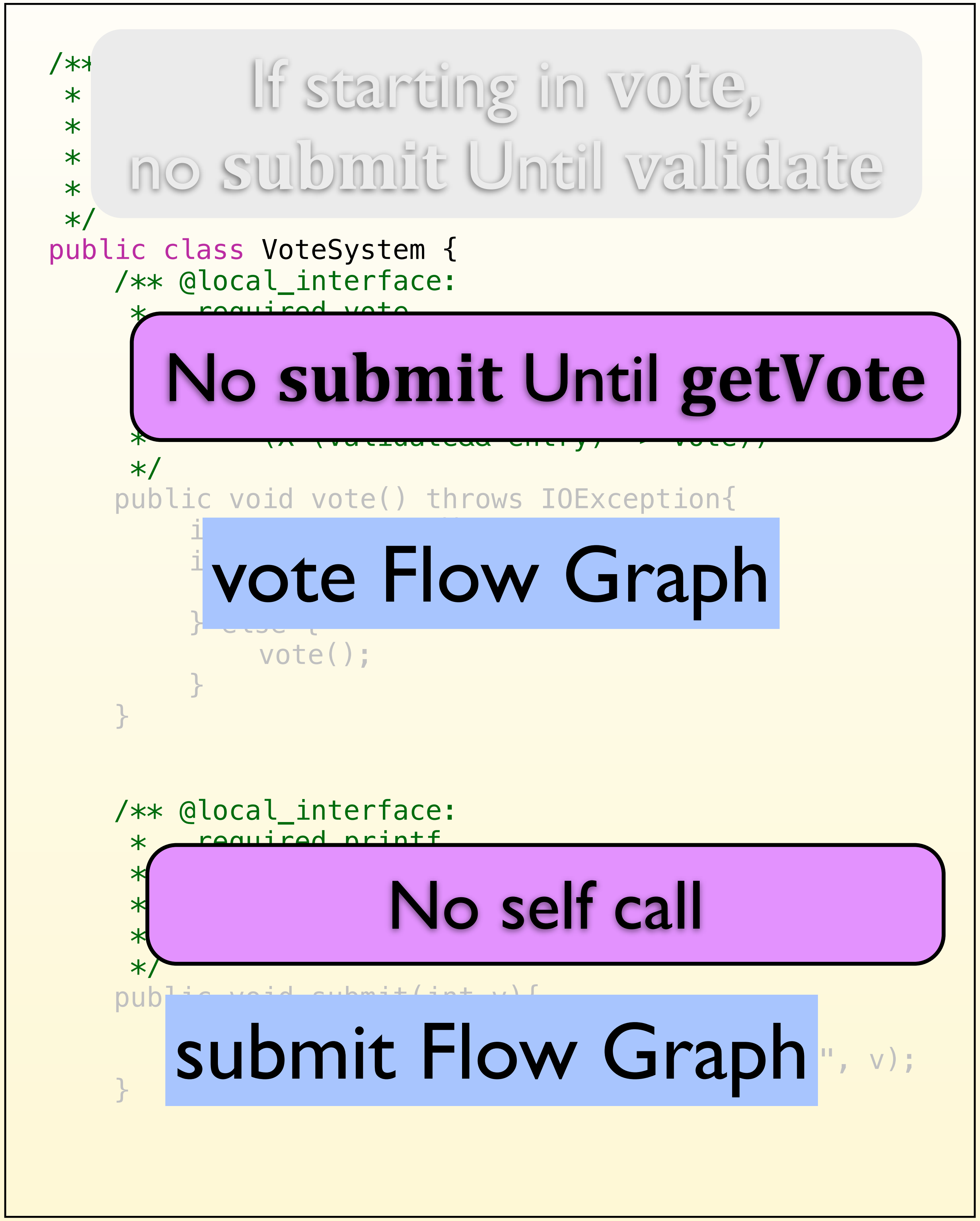
```
    /** @local_interface:
     *    required_validate
     *
     *
     *
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }
```

No self call

```
    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

    }
}
```

No **return** Until **validate**
& no self call

# Task I: Local Check



```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *
     *         (X (validated& entry) -> vote))
     */
    public void vote() throws IOException{
        i
        i
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    public void submit(int v){
                                        ", v);
    }
```

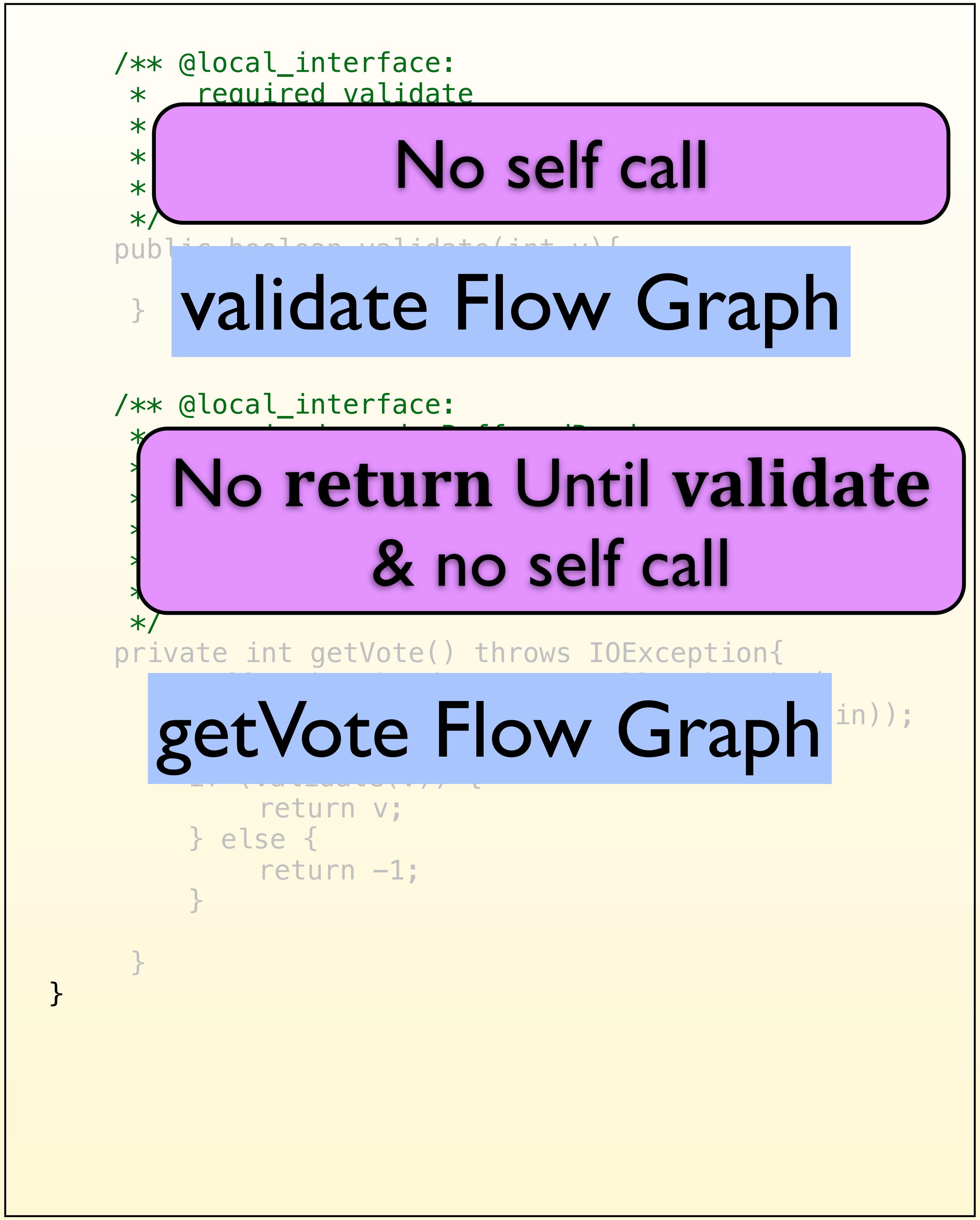
If starting in vote,
no submit Until validate

**No submit** Until **getVote**

vote Flow Graph

No self call

submit Flow Graph



```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
    public boolean validate(int v){

     }

    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
                                                       in));
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

No self call

validate Flow Graph

**No return** Until **validate**
& no self call

getVote Flow Graph

# Task I: Local Check

If starting in vote,
no submit Until validate

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *
     */
    public void vote() throws IOEx
        i
        i
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    public void submit(int v){


    }
```

No submit Unti

vote Flow G

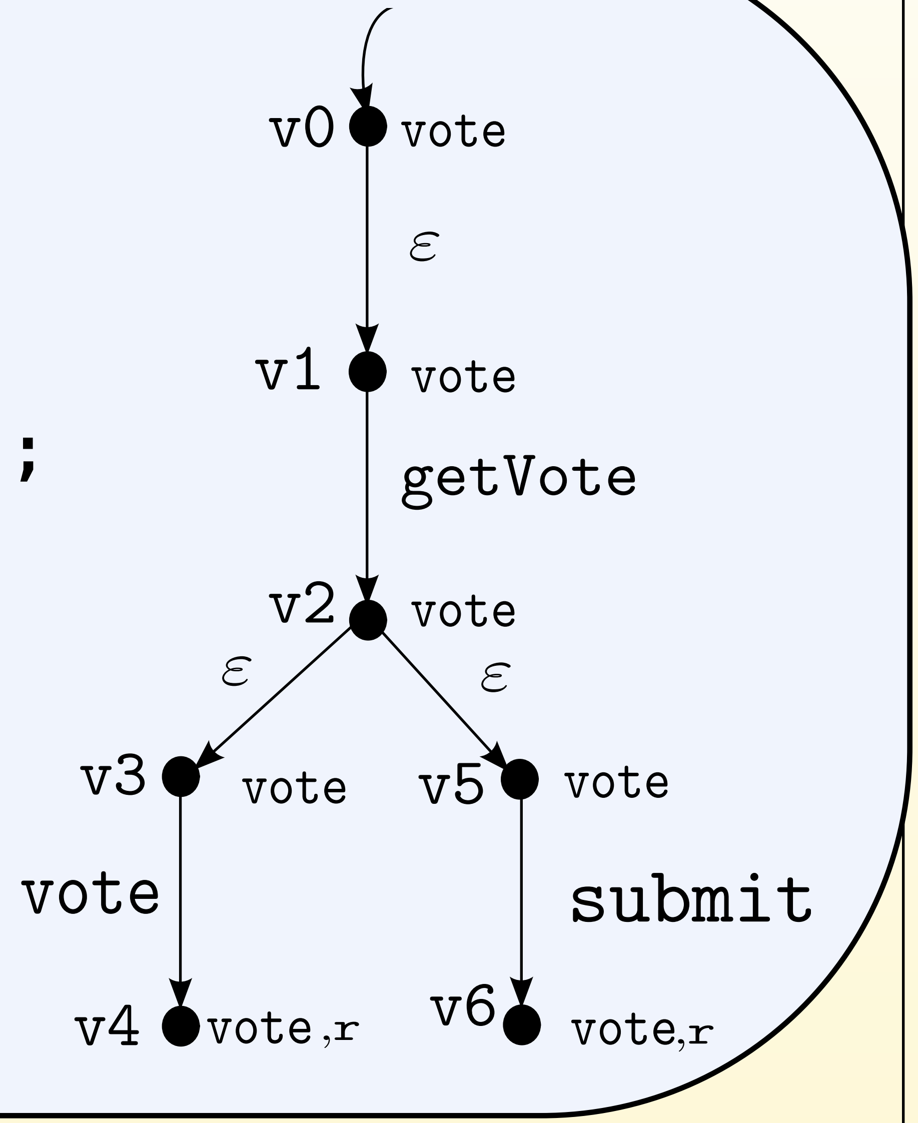
No self

submit Flow Gra

```
/** @local_interface:
 *    required validate
 *
 *
 *
 */
public boolean validate(int v){
```

No self call

```
public void vote()
   throws IOException{
      int v = getVote();
      if (v != -1) {
         submit(v);
      } else {
         vote();
      }
   }
```



# Task 1: Local Check

```
/**
 *
 *
 *
 *
 */
```

If starting in vote,
no submit Until validate

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

**No submit** Until **getVote**

```
     *        (X (validated& entry) -> vote))
     */
    public void vote() throws IOException{
        i
        i
```

vote Flow Graph

```
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    publ
```

No self call

submit Flow Graph

```
                                                  ", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
    publ
```

No self call

validate Flow Graph

```
     }

    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
```

**No return** Until **validate**
& no self call

getVote Flow Graph

```
                                                        in));
            return v;
        } else {
            return -1;
        }

     }
}
```

# Task II: Global Check

```
/**
 *
 *
 *
 *
 */
```

If starting in **vote**,
no **submit** Until **validate**

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No **submit** Until **getVote**

```
     *        (X (validated&& entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

No self call

```
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

No self call

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
```

No **return** Until **validate**
& no self call

```
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }


     }
}
```

# Task II: Global Check

```
/**
 *
 *
 *
 *
 */
```

If starting in **vote**,
no **submit** Until **validate**

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

MaxMod(No **submit**
Until **getVote**)

```
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }


    /** @local_interface:
     *
     *
     *
     *
     *
     *
     *
    p
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

    }
}
```

MaxMod(No **return** Until
**validate** & no self call)

Sunday, May 20, 2012

# Task II: Global Check

```
/**
 *
 *
 *
 *
 */
```

If starting in **vote**, no **submit** Until **validate**

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

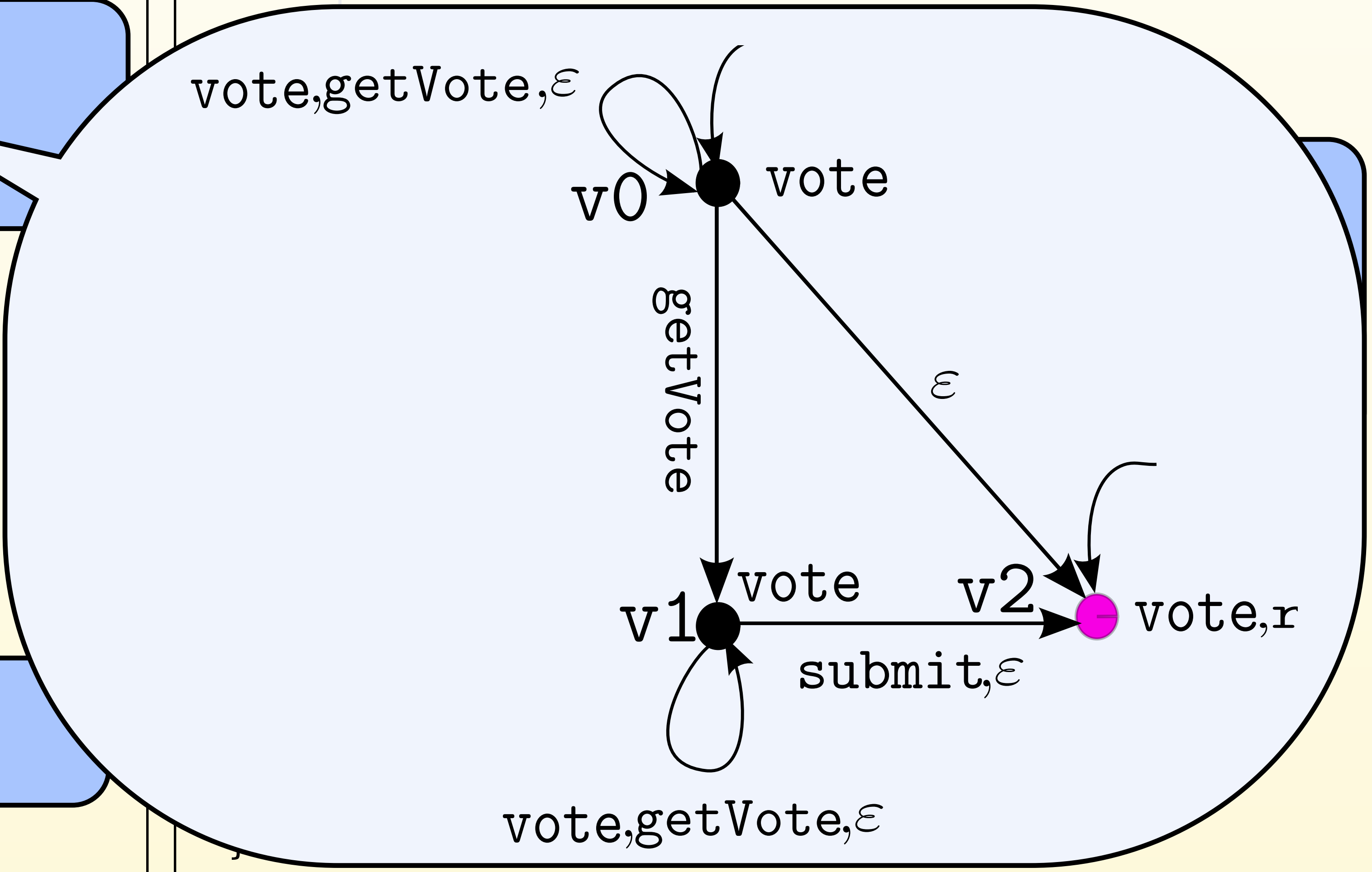
MaxMod(No **submit** Until **getVote**)

```
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

MaxMod(No self call)

```
/** @local_interface:
 *    required validate
 *
 *
 *
 */
public boolean validate(int v){
    return ((1 <= v) && (v <= 5));
}
```

MaxMod(No self call)



# Task II: Global Check

```
/**
 *
 *
 *
 *
 */
```

If starting in **vote**,
no **submit** Until **validate**

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

MaxMod(No **submit**
Until **getVote**)

```
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public void submit(int v){
        System.out.printf(
                "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }


    /** @local_interface:
     *
     *
     *
     *
     *
     *
     *
```

MaxMod(No **return** Until
**validate** & no self call)

```
    p
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }


    }
}
```

# Task II: Global Check

```
/**
 *
 *
 *
 *
 */
```

If starting in **vote**,
no **submit** Until **validate**

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

MaxMod(No **submit** Until **getVote**)

```
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

PDA

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }


    /** @local_interface:
     *
```

MaxMod(No **return** Until **validate** & no self call)

```
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }


    }
}
```

# Task II: Global Check

If starting in **vote**,
no **submit** Until **validate**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

MaxMod(No **submit** Until **getVote**)

```
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public void submit(int v){
        System.out.printf(
                   "The vote %d is submitted!", v);
    }
```

PDA

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }


    /** @local_interface:
     *
```

MaxMod(No **return** Until **validate** & no self call)

```
        BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }


    }
}
```

# Task II: Global Check

**If starting in vote,**
**no submit Until validate**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

MaxMod(No **submit** Until **getVote**)

```
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

PDA

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

MaxMod(No self call)

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }


    /** @local_interface:
     *
     *
     *
```

MaxMod(No **return** Until **validate** & no self call)

```
        ufferedReader br = new BufferedReader(
                new InputStreamReader(System.in));
        nt v = br.read() - 48;
        f (validate(v)) {
            return v;
        } else {
            return -1;
        }

    }
}
```

# Open Systems, Mobile Code

If starting in vote,
no submit Until validate

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No submit Until getVote

```
     *         (X (validated& entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
```

No self call

```
    public void submit(int v){
        System.out.printf(
                "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

No self call

```
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }

    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
        BufferedReader                    redReader(
                                                in));
```

No return Until validate
& no self call

Code not available

```
    }
}
```

Sunday, May 20, 2012

# Open Systems, Mobile Code

If starting in vote,
no submit Until validate

No submit Until getVote

No self call

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *        (X (validated entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
              submit(v);
         } else {
              vote();
         }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    public void submit(int v){
         System.out.printf(
                        "The vote %d is submitted!", v);
    }
```

No self call

No **return** Until **validate**
& no self call

```
     /** @local_interface:
      *    required validate
      *
      *
      *
      */
     public boolean validate(int v){
          return ((1 <= v) && (v <= 5));
      }


     /** @local_interface:
      *
      *
      *
      *
      *
      *
      */
     private int getVote() throws IOException{
          BufferedReader br = new BufferedReader(
                        new InputStreamReader(System.in));
          int v = br.read() - 48;
          if (validate(v)) {
               return v;
          } else {
               return -1;
          }

      }
 }
```

Sunday, May 20, 2012

# Code Evolution

If starting in **vote**,
no **submit** Until **validate**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No **submit** Until **getVote**

```
     *
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

No self call

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }
```

No self call

```
    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

No **return** Until **validate**
& no self call

# Code Evolution

```
/**
 *
 *
 *
 *
 */
```

If starting in vote,
no submit Until validate

```
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No submit Until getVote

```
     *          (X (validated entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
             submit(v);
         } else {
             vote();
         }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     *
     */
```

No self call

```
    public void submit(int v){
         System.out.printf(
                     "The vote %d is submitted!", v);
    }
```

```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
```

No self call

```
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
```

No return Until validate
& no self call

```
    private int getVote() throws IOException{
         BufferedReader b         BufferedReader(
                                             in));
```

Change the Code

```
     }
}
```

# Checking Global Properties

If starting in **vote**,
no **submit** Until **validate**

```
/**
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *     required vote
     *
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }
```

No **submit** Until **getVote**

```
    /** @local_interface:
     *     required printf
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

No self call

```
    /** @local_interface:
     *     required validate
     *
     *
     *
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
    }
```

No self call

```
    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

    }
}
```

No **return** Until **validate**
& no self call

Sunday, May 20, 2012

# Checking Global Properties

If starting in **vote**, never **submit**

```
/*
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
```

No **submit** Until **getVote**

```
     *        (X (validated& entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
```

No self call

```
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                "The vote %d is submitted!", v);
    }
```

```
      /** @local_interface:
       *    required validate
```

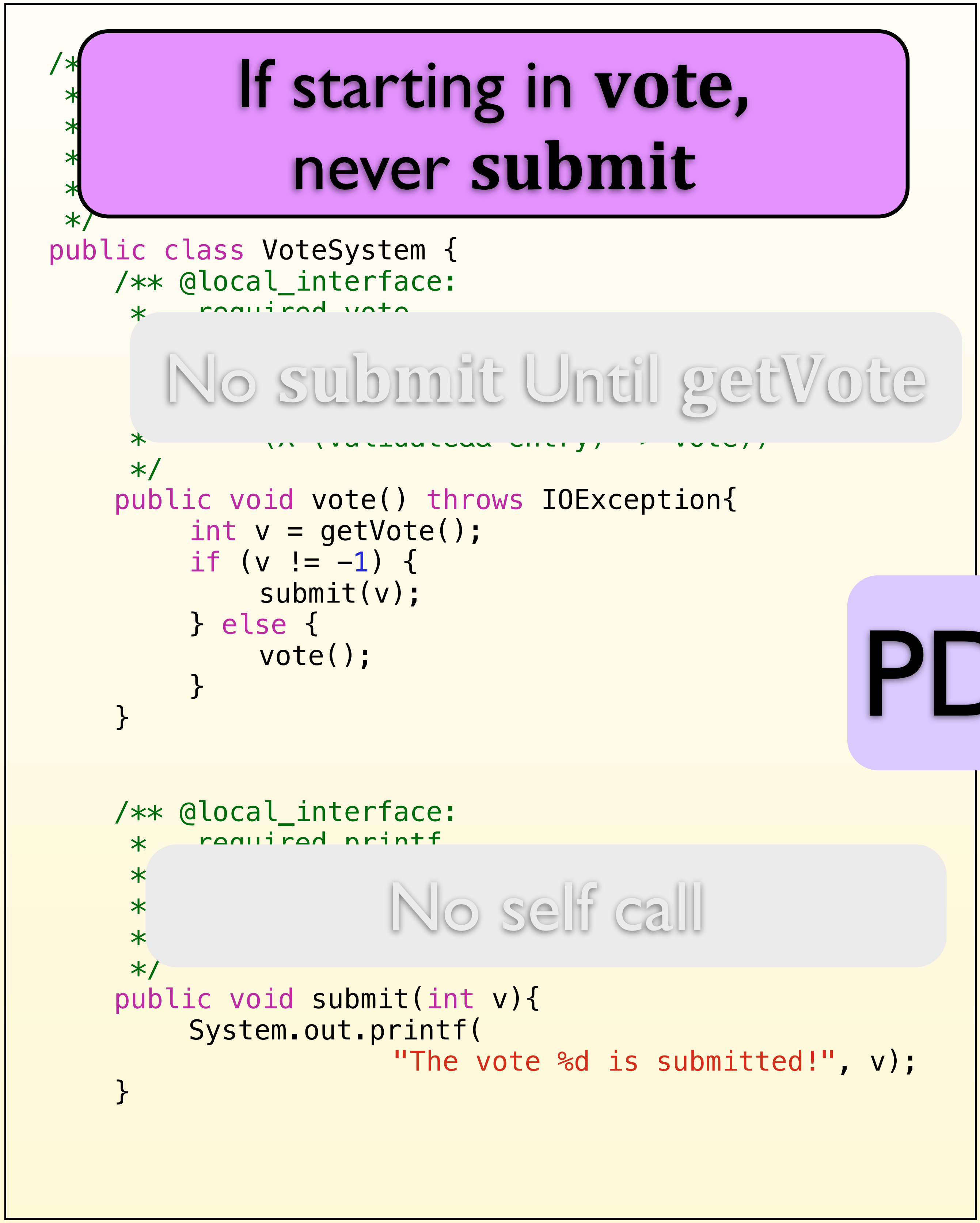
No self call

```
       *
       *
       *
       */
      public boolean validate(int v){
          return ((1 <= v) && (v <= 5));
       }


      /** @local_interface:
```

No **return** Until **validate**
& no self call

```
       */
      private int getVote() throws IOException{
          BufferedReader br = new BufferedReader(
                      new InputStreamReader(System.in));
          int v = br.read() - 48;
          if (validate(v)) {
              return v;
          } else {
              return -1;
          }

       }
 }
```

# Checking Global Properties



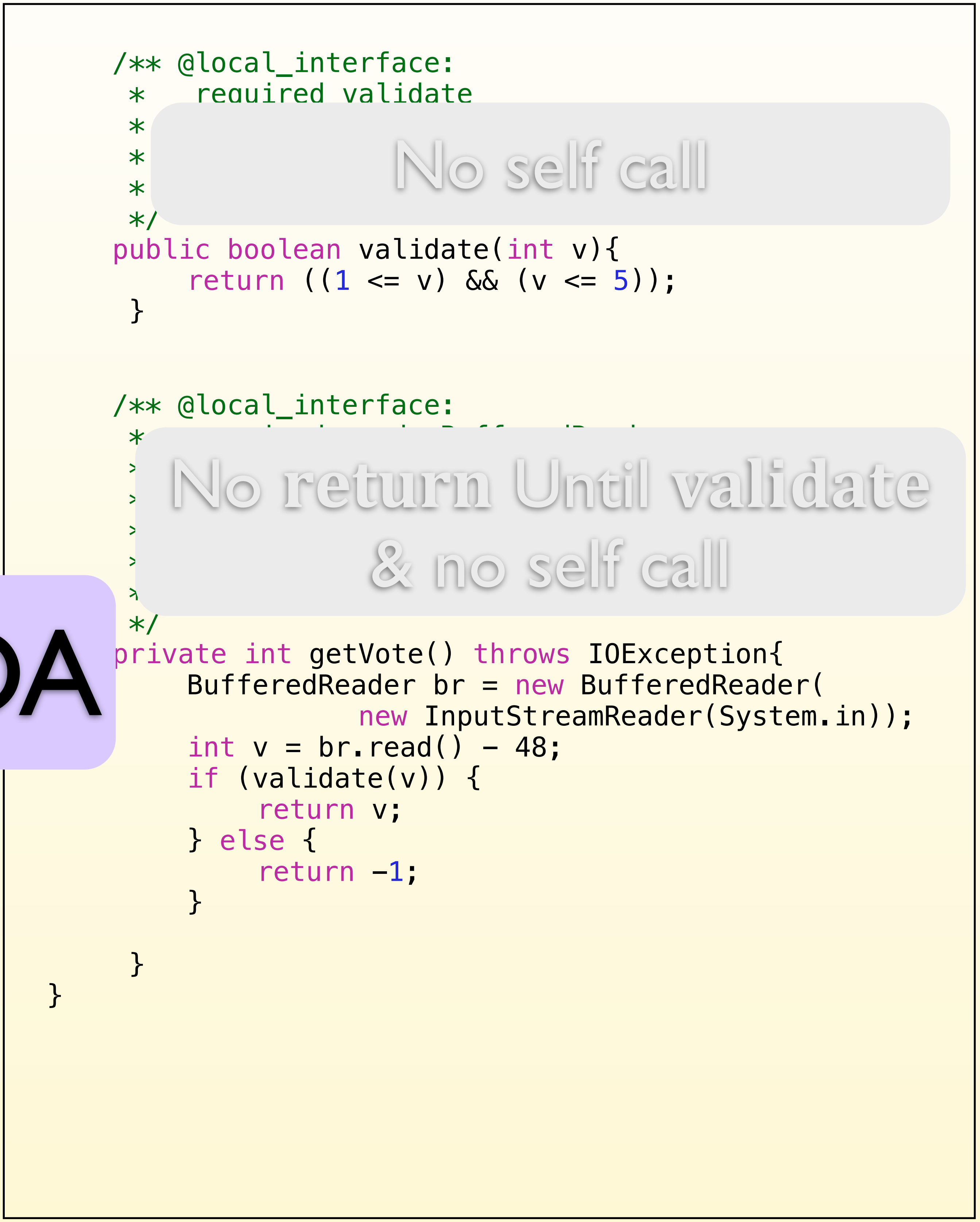
```
/*
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *
     *         (X (validated entry) -> vote))
     */
    public void vote() throws IOException{
        int v = getVote();
        if (v != -1) {
            submit(v);
        } else {
            vote();
        }
    }


    /** @local_interface:
     *    required printf
     *
     *
     *
     */
    public void submit(int v){
        System.out.printf(
                    "The vote %d is submitted!", v);
    }
```

If starting in **vote**, never **submit**

No submit Until getVote

No self call

PDA



```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
    public boolean validate(int v){
        return ((1 <= v) && (v <= 5));
     }


    /** @local_interface:
     *
     *
     *
     *
     *
     *
     */
    private int getVote() throws IOException{
        BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in));
        int v = br.read() - 48;
        if (validate(v)) {
            return v;
        } else {
            return -1;
        }

     }
}
```

No self call

No return Until validate & no self call

# Checking Global Properties

**If starting in vote, never submit**

```
/*
 *
 *
 *
 *
 */
public class VoteSystem {
    /** @local_interface:
     *    required vote
     *       (X (validated entry) -> vote))
     */
    public void vote() throws IOException{
         int v = getVote();
         if (v != -1) {
              submit(v);
         } else {
              vote();
         }
    }

    /** @local_interface:
     *    required printf
     *
     *
     *
     *
     */
    public void submit(int v){
         System.out.printf(
                   "The vote %d is submitted!", v);
    }
```

No submit Until getVote

No self call

PDA

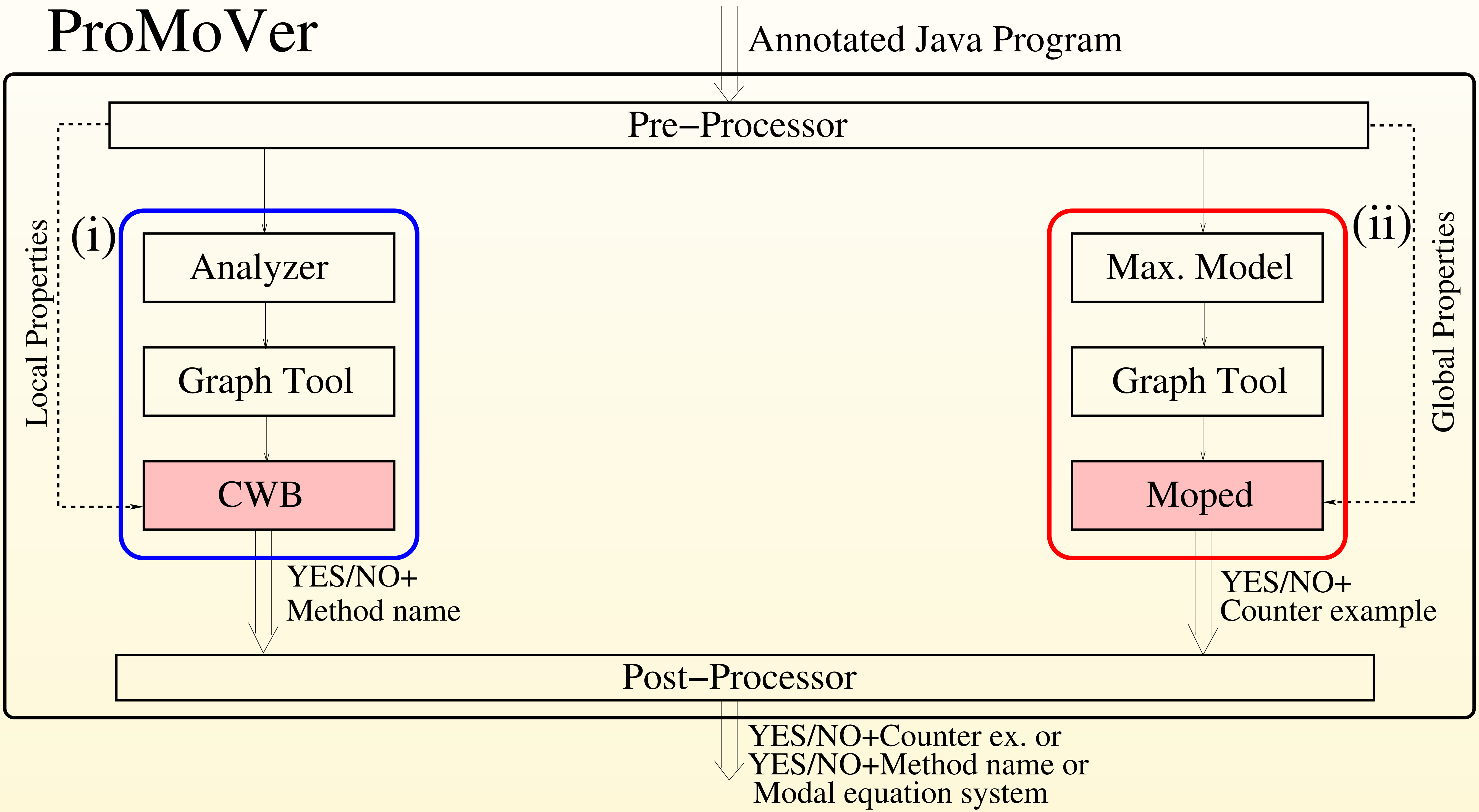
```
    /** @local_interface:
     *    required validate
     *
     *
     *
     */
    public boolean validate(int v){
         return ((1 <= v) && (v <= 5));
     }

    /** @local_interface:
     *
     *
     *
     e int getVote() throws IOException{
     ufferedReader br = new BufferedReader(
                 new InputStreamReader(System.in));
     nt v = br.read() - 48;
     f (validate(v)) {
          return v;
     } else {
          return -1;
     }

    }
}
```

No self call

No return Until validate & no self call

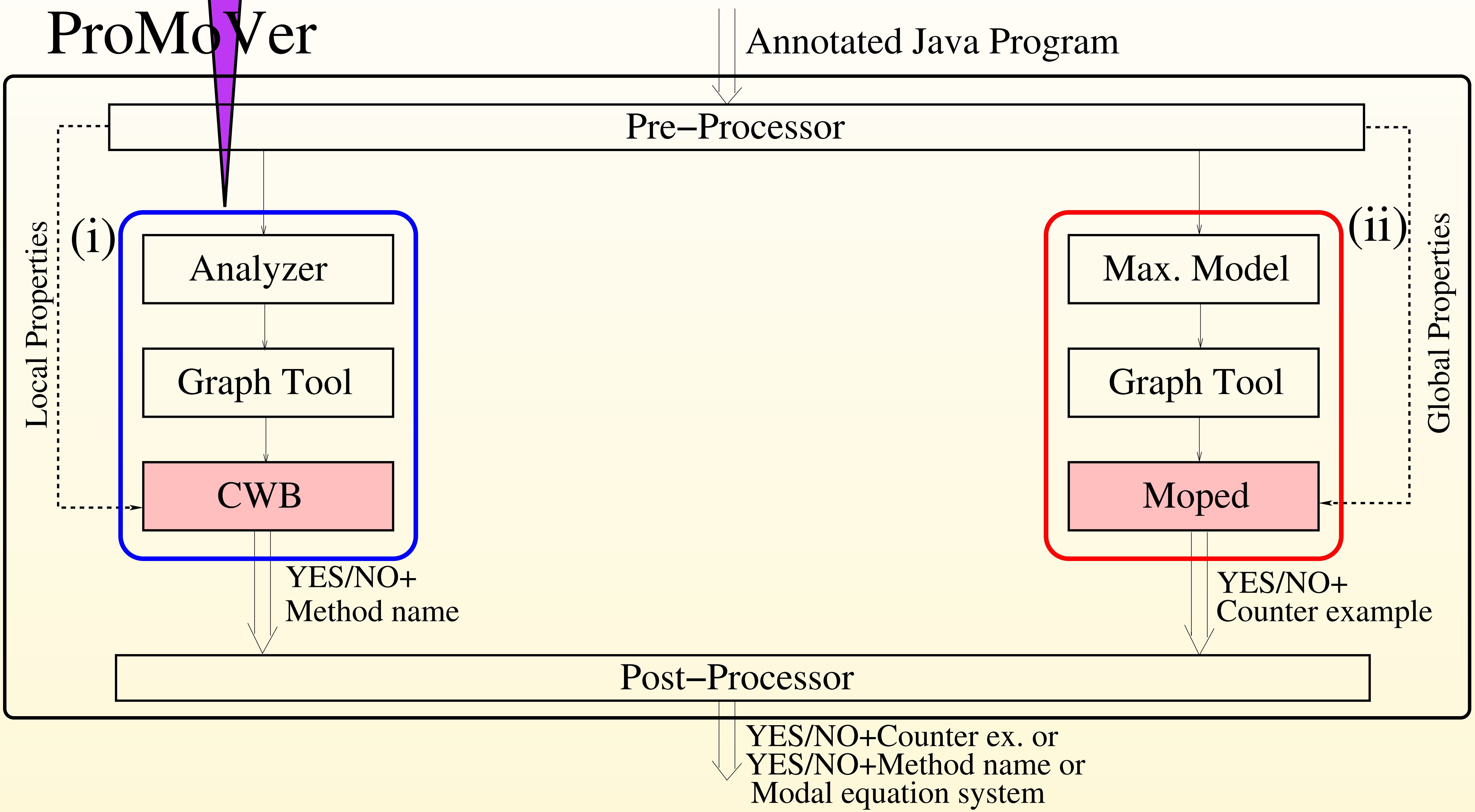
Sunday, May 20, 2012

# ProMoVer

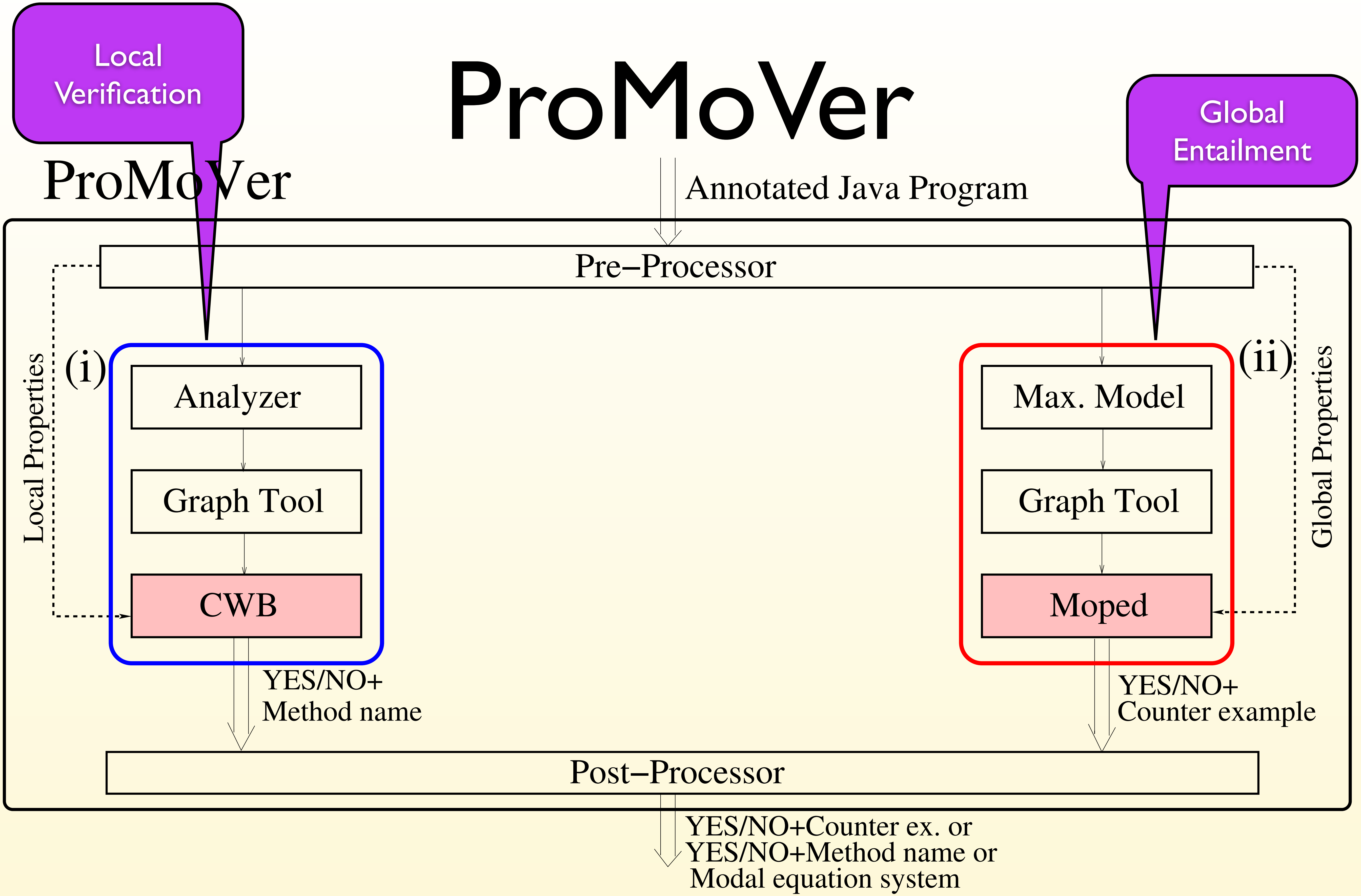

ProMoVer
Annotated Java Program
Pre–Processor
Local Properties
(i)
Analyzer
Graph Tool
CWB
(ii)
Max. Model
Graph Tool
Moped
Global Properties
YES/NO+
Method name
YES/NO+
Counter example
Post–Processor
YES/NO+Counter ex. or
YES/NO+Method name or
Modal equation system

# ProMoVer

Local Verification



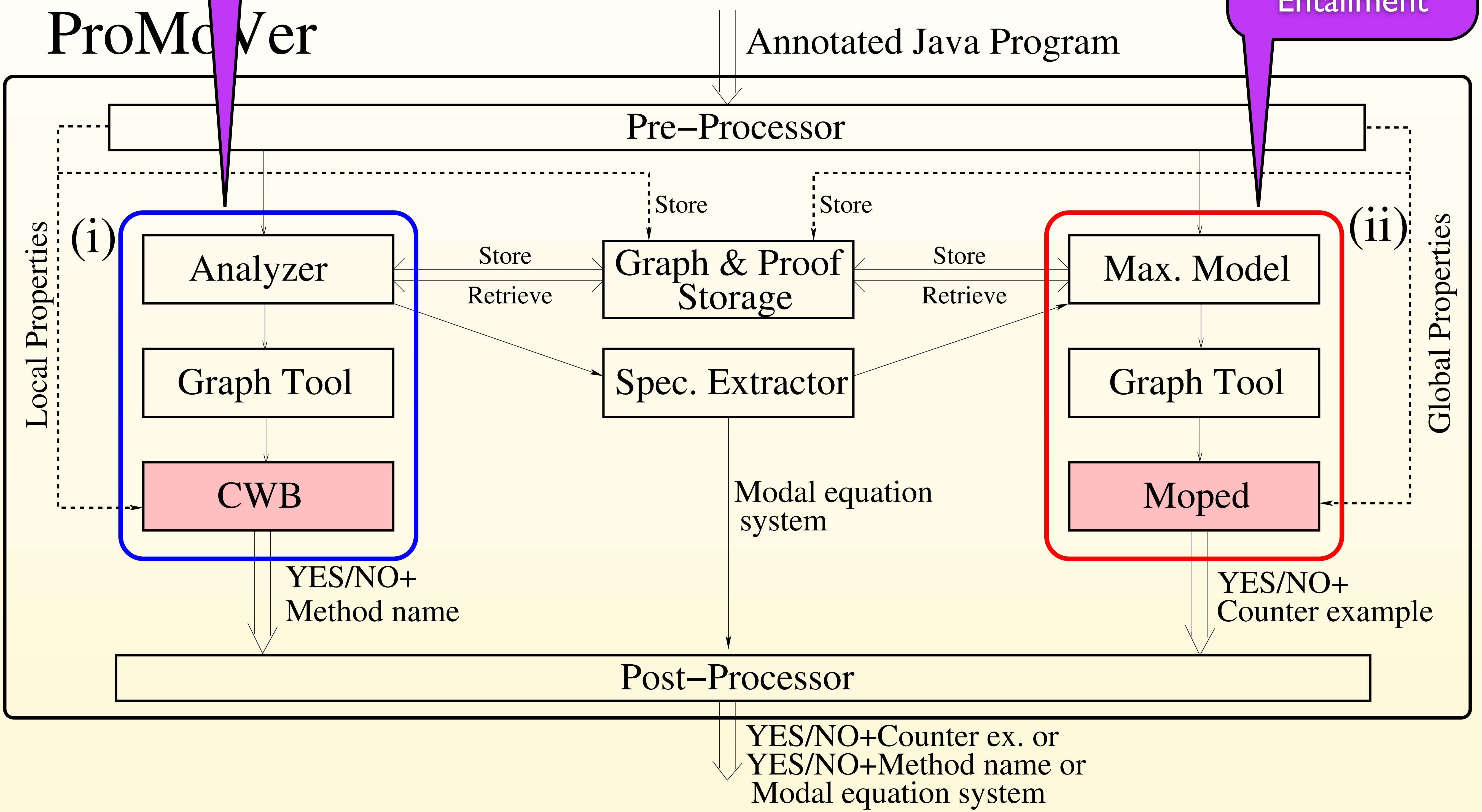
# ProMoVer


Local Verification
Global Entailment
ProMoVer
Annotated Java Program
Pre–Processor
Local Properties
(i)
Analyzer
Graph Tool
CWB
(ii)
Max. Model
Graph Tool
Moped
Global Properties
YES/NO+
Method name
YES/NO+
Counter example
Post–Processor
YES/NO+Counter ex. or
YES/NO+Method name or
Modal equation system

Local Verification

# ProMoVer

Global Entailment



ProMoVer

Annotated Java Program

Pre–Processor

Local Properties

Global Properties

Store

Store

(i)

(ii)

Analyzer

Store

Retrieve

Graph & Proof Storage

Store

Retrieve

Max. Model

Graph Tool

Spec. Extractor

Graph Tool

CWB

Modal equation system

Moped

YES/NO+ Method name

YES/NO+ Counter example

Post–Processor

YES/NO+Counter ex. or
YES/NO+Method name or
Modal equation system

# Demo

# Case Studies

Evaluating ProMoVer with three Java-Card applications

# Case Studies

Evaluating ProMoVer with three Java-Card applications

Global Property

No non-atomic operation **within** a transaction

# Case Studies

Evaluating ProMoVer with three Java-Card applications

Global Property

No non-atomic operation **within** a transaction

| Application | Lines of Code | Local Model Check | Maximal Model Cons. | Global Model Check | Total Time |
|---|---|---|---|---|---|
| AccountAccessor | 190 | 0.5 | 0.7 | 0.9 | 8.7 |
| TransitApplet | 918 | 0.5 | 0.9 | 0.9 | 13.2 |
| JavaPurse | 884 | 0.5 | 13.0 | 1.1 | 22.5 |

# Case Studies

Evaluating ProMoVer with three Java-Card applications

Global Property

No non-atomic operation **within** a transaction

| Application | Lines of Code | Local Model Check | Maximal Model Cons. | Global Model Check | Total Time | Code Change TT | Spec. Change TT |
|---|---|---|---|---|---|---|---|
| AccountAccessor | 190 | 0.5 | 0.7 | 0.9 | 8.7 | 6.0 | 4.6 |
| TransitApplet | 918 | 0.5 | 0.9 | 0.9 | 13.2 | 7.2 | 5.0 |
| JavaPurse | 884 | 0.5 | 13.0 | 1.1 | 22.5 | 9.0 | 5.4 |

# Conclusions

- **ProMoVer**: a completely automated tool for procedure-modular verification
  - algorithmic
  - light weight (Spec. extractor, proof storage & reuse)
  - modular (support open systems, variability)
- ProMoVer is evaluated on real case studies

# Future Work

- Perform a larger case study
- More property languages (patterns,automata)
- Boolean and Object Reference data
    - wider range of properties

# Thank You!