Brief Introduction to ESC/Java

Mads Dam

JML

Adding assertions to Java source code, e.g.
- Preconditions
- Postconditions
- Class invariants

Similar to Eiffel (Design-by-Contract)

Goal: Lightweight, usable by practising programmers

Properties specified as extended Java boolean expressions

JML assertions added as comments (ESC/Java: pragma's) in .java file, between /*@ ... @*/, or after //@

ESC/Java syntax slightly different from JML proper

Pre- and Postconditions

Example:

```
/*@   requires amount >= 0;
      ensures balance == \old(balance) - amount &&
            \result == balance
  @*/
public int debit(int amount)   {
      ...
}
```

`\old(E)`: `E` evaluated in state before method was called

`\result`: The return value

Class Invariants

Class invariants must always be preserved

```
public class Wallet  {
   public static final short MAX_BALANCE = 1000 ;
   private short balance;
   /*@ invariant 0 <= balance
        && balance <= MAX_BALANCE
        @*/
        ...
```

```
Invariants must be
   – Preserved by all methods,
     i.e. implicitly included in both pre- and postcondition of
     methods, including exceptional termination
   – Established by all constructors,
     i.e. implicitly included in postconditions of constructors
```

Other Pragmas

Introducing assumptions:
```
/*@ assume balance >= 0 @*/
```

Exceptional postconditions:
```
/*@    requires amount >= 0 ;
       ensures true ;
       exsures (SomeException) balance >= 0  @*/
public int debit(int amount) throws SomeException
   ...
```

Only SomeException can be thrown
Whenever SomeException is thrown, balance >= 0

Assertions:
```
/*@ assert balance >=0 @*/
```

See ESC/Java manuals for more pragmas

ESC/Java

Extended static checker by Leino et al, Compaq
- Checks JML annotated Java code
- Unsound
  - Annotations might be wrong, but ESC/Java does not identify a problem
- Incomplete
  - ESC/Java might report an error, even if no error is actually present
- Good at routine checks of relatively simple properties
  - Like: Absence of runtime exceptions
- Bad at loops
  - Loops only traversed once

ESC/Java Benefits

- ESC/Java forces important properties to be noted and recorded
- Often the properties are obvious, if you understand the code
- But for larger programs, who has complete understanding of everything?
- If you have the important properties properly noted then
  - understanding
  - maintenance

  is much easier

ESC/Java Limitations

Typical example:

```
for (i = 0; i < buffer.length ; i++)
   {     ...       }
//@ assert i == buffer.length ;
...
```

Not provable in ESC/Java!