DD2385
Programutvecklingsteknik
Några bilder till föreläsning 1
24/3 – 2014

Innehåll

- Kursöversikt
- Javarepetition/Javaintroduktion
- UML - klassdiagram-introduktion
  i anslutning till Java-exemplen

Kursmål, förkortat

- Använda objektorienterade tekniker vid eget programmeringsarbete
- Redogöra för och tillämpa kriterier för god objektorienterad design
- Redogöra för de vanligaste designmönstren inom objektorienterad programutveckling samt välja lämpliga mönster för enkla tillämpningsexempel.
- Använda UML-klassdiagram för att på ett överskådligt och tydligt sätt planera och dokumentera eget programmeringsarbete.
- Läsa och förstå UML-klassdiagram, t.ex. som introduktion till nya designmönster.
- Använda Javas bibliotheksklasser och ramverk

Kursinnehåll

- Processen från Informell kravspecifikation
  till Färdigt program
- Objektorienterad programmering i Java med
  UML, designkriterier, designmönster

- Processen från en informell kravspecifikation till färdigt program
  - OOA (ObjektOrienterad Analys)
  - OOD (ObjektOrienterad Design)
  - Designmönster
  - Kriterier för god design
  - Refactoring
  - UML–klassdiagram
  - UML–diagram av andra slag
  - Utvecklingsmetodik
    - Traditionell metodik
    - "Agile methodologies", t.ex. extremprogrammering (XP), SCRUM
  - Testning
  - Versionshantering
- Objektorienterad programmering i Java med
  - Klasser, objekt, konstruktorer, arv, static
  - Abstrakta klasser, interface
  - Klassbibliotek och ramverk
  - Designmönster i Java-biblioteken
  - Grafiska gränssnitt, i första hand med Swing
    - Enkla komponenter, t.ex. JLabel, JButton
    - Avancerade komponenter, t.ex. JList, JTree, JEditorPane
  - Parallella processer: trådar
  - Klient-server-program (där klient och server är på olika datorer)
- XML
  - "Riktig" XML
  - XML-liknande notation

Java-begrepp att ta upp idag

- Klass, objekt
- Instansvariabel, instansmetod
- Klassvariabel, klassmetod
- Konstruktor
- Typer, typdeklarationer
- Arv, konstruktor vid arv
- Metoden `public String toString()`
- Metoden `public static void main(String[] a)`
- Grundläggande syntax

Exemplen med
Spelkort och Patienskort
är mycket viktiga.
Studera dem noga!

Klass för Spelkort

```
class Spelkort {
  String farg;    int valor;

  //Konstruktor
  Spelkort (String f, int v) {
    farg = f;   valor = v;
  }

  //Visa text-utseende
  public String toString () {...}

/***********************************
* Ovan:  objektmallen for Spelkort  *
* Nedan: klassen som behallare      *
***********************************/

  static String[] specvalor
      = {"ESS", "KNEKT", "DAM", "KUNG"};
  // main - metod skulle kunna ligga har
}
```

toString-metoden i Spelkort

```
class Spelkort {

  // instansvariabler, konstruktor
  // som visats tidigare

  public String toString () {
     String valorString;
     if (valor == 1)
        valorString = specvalor[0];
     else if (valor >=2 && valor <=10)
        valorString = "" + valor;
     else
        valorString = specvalor[valor - 10];
     return farg + "␣" + valorString;
  }
  // ... som tidigare
}
```

toString-metoden i Spelkort (samma som förra bilden)

instansvariabel, klassvariabel, lokal variabel

```
class Spelkort {
   // instansvariabler, konstruktor
   // som visats tidigare
   public String toString () {
      String valorString;
      if (valor == 1)
         valorString = specvalor[0];
      else if (valor >=2 && valor <=10)
         valorString = "" + valor;
      else
         valorString = specvalor[valor - 10];
      return farg + "␣" + valorString;
   }
   // ... som tidigare
}
```

testa Spelkortsklassen:

Klass med bara main-metod som skapar och visar kortlek

```
class TestaSpelkort {
  public static void main (String[] arg) {

    // Skapa vektor med plats for 52 Spelkort

    // Skapa korten

    // Visa korten
  }
}
```

testa Spelkortsklassen

```
class TestaSpelkort {
  public static void main (String[] arg) {
    String[] farger =
      {"Hjarter", "Spader", "Ruter", "Klover"};
    // Skapa vektor med plats for 52 Spelkort.
    Spelkort[] kortlek = new Spelkort[52];

    // Skapa korten
    int kortnr = 0;
    for (String farg : farger)
      for (int valor = 1; valor<=13; valor++)
        kortlek[kortnr++] =
          new Spelkort(farg, valor);

    // Visa korten
    for (Spelkort spk : kortlek)
      System.out.println(spk);
  }
}
```

Kör testprogrammet!

Filer: Spelkort.java TestaSpelkort.java

Kompilera:
>>>javac Spelkort.java TestaSpelkort.java
=> Spelkort.class TestaSpelkort.class

Kör programmet:
>>>java TestaSpelkort (klass med main-metod)

Utskrift i terminalfönstret ...

```
Hjärter ESS
Hjärter 2
Hjärter 3
Hjärter 4
Hjärter 5
Hjärter 6
Hjärter 7
Hjärter 8
Hjärter 9
Hjärter 10
Hjärter KNEKT
Hjärter DAM
Hjärter KUNG
Spader ESS
Spader 2
Spader 3
Spader 4
Spader 5
Spader 6
Spader 7
Spader 8
Spader 9
Spader 10
Spader KNEKT
Spader DAM
Spader KUNG
```

```
Ruter ESS
Ruter 2
Ruter 3
Ruter 4
Ruter 5
Ruter 6
Ruter 7
Ruter 8
Ruter 9
Ruter 10
Ruter KNEKT
Ruter DAM
Ruter KUNG
Klöver ESS
Klöver 2
Klöver 3
Klöver 4
Klöver 5
Klöver 6
Klöver 7
Klöver 8
Klöver 9
Klöver 10
Klöver KNEKT
Klöver DAM
Klöver KUNG
```

. . . fast det blir bara en kolumn

Patienskort ärver från Spelkort

Korten har framsidan eller baksidan "synlig".

```
class Patienskort extends Spelkort {

   boolean rattvand;

   Patienskort (String f, int v, boolean rv) {
      super(f,v);
      rattvand = rv;
   }

   void vand () {
      rattvand = !rattvand;
   }

   public String toString() {...}
}
```

toString-metoden for Patienskort

```
public String toString() {
    if (rattvand)
        return super.toString();
    else
        return "BAKSIDA";
 }
```

```
super.toString();
```

$\Longrightarrow$

```
toString()
```

i superklassen `Spelkort` anropas

Litet kodexempel med Patienskort

```
Patienskort pk =
        new Patienskort("Spader",12,true);
System.out.println(pk);
pk.vand();
System.out.println(pk);
pk.vand();
System.out.println(pk);
```

Första utskriften ger `Spader DAM`

Andra utskriften ger `BAKSIDA`

Tredje utskriften ger `Spader DAM` igen

testa Patienskortsklassen:

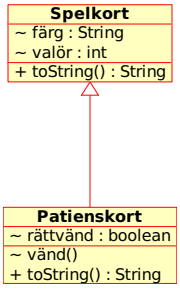
```
class TestaPatienskort {
  public static void main (String[] arg) {
    String[] farger =
      {"Hjarter", "Spader", "Ruter", "Klover"};
    // Skapa vektor med plats for 52 Patienskort.
    Patienskort[] kortlek = new Patienskort[52];

    // Skapa korten
    int kortnr = 0;
    for (String farg : farger)
      for (int valor = 1; valor<=13; valor++)
        kortlek[kortnr++] =
          new Patienskort(farg, valor, true);

    // Visa korten
    for (Patienskort pk : kortlek)
      System.out.println(pk);
  }
}
```

UML

Standard för grafiska beskrivningar av olika aspekter av objektorienterade program.

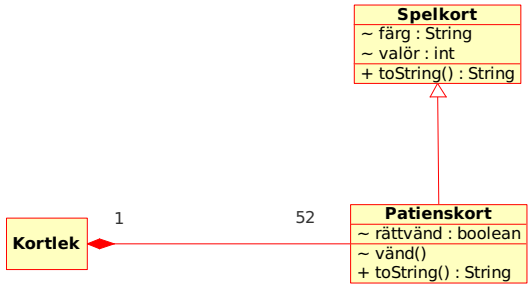
- class diagram
- use case diagram
- statechart diagram
- sequence diagram
- activity diagram
- collaboration diagram
- component diagram
- deployment diagram

UML för Spelkort och Patienskort




```
class Patienskort extends Spelkort {
    ...
}
```

UML för Kortlek, Spelkort och Patienskort




```
class Kortlek {
  Patienskort[] kortlek = new PatiensKort[52];
    ...
}
```

Klassen Kortlek

```
class Kortlek {
    Patienskort[] lek = new PatiensKort[52];

    Kortlek () {
        // Skapa 52 kort som vi gjorde tidigare
        // i testprogrammet
    }

    void blanda () { ... }  // ej i
                            // UML-diagrammet!
    // fler metoder
}
```