Åtkomst och inkapsling

Behåll dina hemligheter:
Ändra ditt kontrakt

Bakgrund (1)

jonkv@ida

- Vi vill skriva klassen **SortedList**
  - Ska garantera att elementen är i **sorterad ordning**!

```
class TestSortedList
{
   public static void main(String[] args) {
      SortedList letters = new SortedList();
      letters.add("A");
      letters.add("Z");
      System.out.println(letters);
      // [ "A", "Z" ]

      letters.add("M");
      System.out.println(letters);
      // [ "A", "M", "Z" ]
   }
}
```

Bakgrund (2)

- **SortedList**: En partiell implementation

```
class SortedList {
   String[] elements;
   int      size;

   String   get(int pos) { return ... }

   void     add(String el) {
      int pos = findNewPositionFor(el);
      insertAt(pos, el);
   }

   int      findNewPositionFor(String el) { ... }
   void     insertAt(int pos, String el) { ... }
}
```

Lagring för alla element

Hämtar element

Stoppar in element på rätt plats…

… med dessa hjälpmetoder (nya begrepp!)

Att gömma information

jonkv@ida

- Ofta vill vi ”gömma” delar av klasser och objekt – varför?

```
class SortedList {
   String[] elements;
   int      size;

   String   get(int pos) { return ... }

   void     add(String el) {
      int pos = findNewPositionFor(el);
      insertAt(pos, el);
   }

   int      findNewPositionFor(String el) { ... }
   void     insertAt(int pos, String el) { ... }
}
```

Gömma: För att inte förvirra användarna

jonkv@ida

- Mindre synligt ➔ mer överskådligt

```
class SortedList {
   String[] elements;
   int      size;

   String   get(int pos) { return ... }

   void     add(String el) {
      int pos = findNewPositionFor(el);
      insertAt(pos, el);
   }

   int      findNewPositionFor(String el) { ... }
   void     insertAt(int pos, String el) { ... }
}
```

Användaren **behöver** bara detta!

Att det andra syns **förvirrar**, gör klassen oöverskådlig!

Gömma: Minska åtaganden (kontrakt)

jonkv@ida

- Det vi har **visat upp**, måste vi normalt **ha kvar**

```
class SortedList {
   String[] elements;
   int      size;

   String   get(int pos) { return ... }

   void     add(String el) {
      int pos = findNewPositionFor(el);
      insertAt(pos, el);
   }

   int      findNewPositionFor(String el) { ... }
   void     insertAt(int pos, String el) { ... }
}
```

Visar vi detta, blir det en del av **kontraktet**

Behövs inte!

Göm det ➔
enbart get/add används ➔
vi kan **byta ut** resten
om vi vill ➔ frihet!

Att gömma data i Java

jonkv@ida

- Många OO-språk tillåter oss att **gömma data**
  - I Java kan klasser, gränssnitt, fält och metoder vara:
    - **public** – kod i **alla** klasser har tillgång
    - **protected** – tillgång i **underklasser** + andra i samma **paket**
    - [*inget*] – tillgång i samma **paket** (bör undvikas)
    - **private** – tillgång inom samma **klass**

Vad hade private betytt utan OO?

```
public class SortedList {
   private String[]   elements;
   private int        size;

   public String      get(int pos) { return ... }
   public void        add(String el) {
      int pos = findNewPositionFor(el);
      insertAt(pos, el);
   }
   private int        findNewPositionFor(String el) { ... }
   private void       insertAt(int pos, String el) { ... }
}
```

Göm all info om **implementationsdetaljer** → Vi lovar mindre i vårt kontrakt!

Skapa ett **stabilt** gränssnitt för "allmänheten": get(), add()

Gömma: Minska åtaganden (kontrakt)

jonkv@ida

- Det som **ingen annan ser** kan vi **ändra på**

```
public class SortedList {
   private String[]   elements;
   private int        size;

   public String      get(int pos)      { return ... }
   public void        add(String el)    {
      int pos = findNe
      insertAt(pos, el);
   }
   private int        f
   private void       i
}
```

```
public class SortedList {
   private ListNode   first;
   private int        size;

   public String      get(int pos)      { return ... }
   public void        add(String el)    {
      ListNode pos = findNewPositionFor(el);
      insertElementAt(pos, el);
   }
   private ListNode   findNewPositionFor(String el) { ... }
   private void       insertElementAt(ListNode pos, String el) { ... }
}
```

Den "publika" delen har kvar **samma signatur** (metodnamn, parametrar, ...)

Gömma: För att uppfylla vårt kontrakt

jonkv@ida

```
/**
 * This class maintains a sorted list of elements.
 * Elements will always be in sorted order,
 * regardless of the order in which they are added.
 */
public class SortedList {
   private String[]     elements;
   private int          size;

   public String   get(int pos)       { return ... }
   public void     add(String el)     {
      int pos = findNewPositionFor(el);
      insertAt(pos, el);
   }
   private int     findNewPositionFor(String el) { ... }
   private void    insertAt(int pos, String el) { ... }
}
```

Om användare kunde anropa insertAt med fel position blir listan osorterad…

Då **bryter** vi vårt **löfte**!

Två betydelser hos…

Inkapsling (Encapsulation)

Koppla ihop data och funktioner

- Fundamentalt i OO
- "Objekt: en **kapsel** som innehåller både data och funktioner"

Begränsa tillgång till medlemmar

- Genom accessmodifierare
- "Vissa medlemmar är instängda i en **ogenomskinlig kapsel**, och kan inte ses eller utnytttjas från utsidan"

Privata fält

jonkv@ida

- Vanlig princip: **Fält** är implementationsdetaljer, **ska vara privata**
  - **Om** det verkligen är rimligt att andra klasser ska komma åt fältvärden:
    - Skapa en public **accessor**-metod (**getter**) vid behov
      Skapa en public **mutator**-metod (**setter**) vid behov

Kan ändra intern representation:
Ändra bara getter/setter, som är i samma klass

- ```
  private double diameter;
  /** Set the radius to r (must not be negative). */
  public void setRadius(final double radius) {
     if (r >= 0.0) diameter = 2 * radius;
     else throw new IllegalArgumentException("Negative radius: " + r);
  }
  ```

Kan lägga till konsistenskontroll
  ```

Namngivning: Getter, Setter

jonkv@ida

- Namngivning:

```
public class Circle {
   private double radius;
   private boolean visible;

   /** Set the radius to r (must not be negative). */
   public void setRadius(final double r) {
      if (radius >= 0.0) radius = r;
      else throw new …;
   }

   public double getRadius() {
      return radius;
   }
   public boolean isVisible() {
      return visible;
   }
}
```

Setters:
void setProperty(…)

Getters:
getProperty()

Booleska getters:
isProperty()

Mer info: http://docs.oracle.com/javase/tutorial/javabeans/writing/properties.html

Tillåtna publika fält

- Vissa anser att det finns **undantag**
  - ”Rena” datastrukturer
    (inte mycket beteende, osannolikt att det skulle ändras i framtiden)

```
public class Dimension {
    public int width;
    public int height;
}
```