

Programmering 2

Java - del 7

Interface

Enumeration

Interfaces

- Med interface betecknas allmänt de resurser en klass (eller ett objekt) ställer till förfogande
- Mängden resurser som kan användas av “andra”
- I Java formaliseras detta med `interface`-konstruktionen
- Ett Java `interface` är en mängd konstanter och abstrakta metoder
- En interface lägger på så sätt fast “krav” på de klasser som implementerar interfacet

```
class klassnamn implements interfacenamn {  
    ...  
}
```

OBS! Det får finnas flera

Interface - exempel

Finns i paketet java.lang

```
public interface Comparable<T> {  
    int compareTo(T obj);  
}
```

Comparable i Shapes2 exemplet

```
public abstract class Figure implements Comparable<Figure> {
    private Position position;
    private Color color;
    ...
    public int compareTo(Figure obj) {
        if (this.getArea() < obj.getArea() )
            return -1;
        else if (this.getArea() > obj.getArea() )
            return 1;
        else return 0;
    }
    public abstract double getArea();
    // Compute and return area of figure
}
```

För att ha något att jämföra

Ny Square

```
import java.awt.Rectangle;  
  
public class Square extends Figure {  
    ...  
    public double getArea() {  
        return size*size;  
    }  
}
```

Interface - deklaration

```
public interface Interfacename {  
    • konstantdeklarationer  
    • abstrakta metodhuvuden  
}
```

Interfaces

- Varje interface-implementation är en ny “version” av interfacet
- Interface är inga klasser och kan inte instantieras
- En klass kan ärva från en klass och dessutom implementera ett eller flera interface
- Interface konstanterna är tillgängliga i den implementerade klassen
- En interface kan “ärva” från ett eller flera interface (m h a extends)
- En klass som implementerar ett sådan interface måste också implementera alla ärvda metoder

Interfaces

- Varje interface definierar en klasstyp
- Interface liknar en superklass för alla klasser som implementerar interfacet
 - ➔ Polymorfi gäller även för interface
 - ➔ Referenser får referera till objekt av en subklass
- Interface-konceptet har många likheter med (multipelt) arv, dock utan dess nackdelar
- Multipelt arv finns inte i Java

Interface

- Snabba att implementera
- Bryter beroenden
- Gör det enklare att samarbeta
- Gör det enkelt att i ett senare skede byta ut implementationer mot effektivare

Skriv generell kod

- Skriv alltid kod så att den använder sig av klasser/interface så högt upp i hierarkin som möjligt
- Ska ni bara göra saker som finns definierade i Figure-klassen så använd er då av en Figure-referens i stället för tex en mer specifik Triangle, så får ni mer generell kod

```
moveFigure(new Triangle())  
void moveFigure(Figure f) {  
    f.moveVertical(10);  
    f.moveHorizontal(10)  
}
```

Enumeration

- Skandal att detta inte fanns i Java 1.0
- Uppräkningsbar typ
- Man räknar upp alla värden som typen ska ha
- Tidigare innan detta dök upp i java användes heltalskonstanter ofta för uppgifter där man nu använder enums

Med enum

```
public class Navigate {  
    public enum Direction { NORTH, SOUTH, EAST, WEST };  
  
    void go( Direction dir ) {  
        switch( dir ){  
            case NORTH:  
                // ...  
                break;  
            case SOUTH:  
                // ...  
                break;  
            case WEST:  
                // ...  
                break;  
            case EAST:  
                // ...  
                break;  
        }  
    }  
}
```