Föreläsning 8
Objektorienterad programmering
DD1332

- Reguljära uttryck
- List, ArrayList
- Generics
- Träd
- Trädtraversering

1

Reguljära uttryck

Ett sätt att beskriva en mängd av strängar

- Detta görs m.h.a paketet java.util.regex
- Meta tecken: <([{\^-=$!|})?*+.>

Exempel	Betydelse
t+	Ett eller flera t
t*	Inget eller flera t
t?	Ett eller inget t
t{23}	Exakt 23 efterföljande t
t{23,}	Minst 23 efterföljande t
t{2, 7}	Minst 2 och inte fler än 7 efterföljande t

2

Kodexempel

```
…
while (true) {
    Pattern pattern = Pattern.compile(sc.nextLine());
    Matcher matcher = pattern.matcher(sc.nextLine());
    System.out.println(matcher.groupCount());
    while (matcher.find()){
        String text=matcher.group();
        int start=matcher.start();
        int slut=matcher.end();
        System.out.println(text+" "+start+" "+slut);
   }
}
```

3

List

Klasser för att lagra objekt

- ArrayList: effektiv när det gäller iteration, men inte effektiv när man ofta vill lägga till eller ta bort objekt.
- Vector: Samma som ArrayList fast metoderna är synchronized för trådsäkerhet, och därmed mindre effektiv än ArrayList
- LinkedList: är som ArrayList med den skillnaden att objekten är dubbellänkade till varandra och därmed tillkommer metoder som gör att klassen är mer användbar för att bygga Stacker och Köer.

4

ArrayList

```
import java.util.*;
…
public static void main(String[] args){
    ArrayList allt=new ArrayList();
    for (int i=0; i<1000; i++){
        allt.add(new Bil())
    }

    for(int j=0; j<allt.size();j++){
        Bil b = (Bil) allt.get(j);
        System.out.println(b);
    }
}
```

5

Sortera ArrayList

- For sortering kan man använda klassmetoden `sort` i klassen `Collections`, den kräver dock att objekten är implementerad `Comparable`

6

Exempel

```
import java.util.*;
…
public static void main(String[] args){
    String[] sa={"one", "two", "three", "four"};
    ArrayList allt = new ArrayList();
    for (int i=0; i<1000; i++){
        allt.add(sa)
    }
    Collections.sort(allt);
    for(int j=0; j<allt.size();j++){
        String s = (String) allt.get(j);
        System.out.println(s);
    }
```

7

Generiska typer

- I listor kan man lagra objekt av olika slag, men det är inte ofta man gör detta utan oftast är det objekt av samma slag som man lagrar i en lista.
- Generiska typer möjliggör en säkrare list-typ där man får kompileringsfel istället för runtimesfel om man försöker lagrar fel typ av objekt i listan.

8

Exempel

```
import java.util.*;
…
public static void main(String[] args){
    ArrayList<Bil> allt=new ArrayList<Bil>();
    for (int i=0; i<1000; i++){
        allt.add(new Bil())
    }

    for(int j=0; j<allt.size();j++){
        Bil b = allt.get(j);
        System.out.println(b);
    }
}
```

9

Generiska typer och ärv

```
class A{}
class B axtends A{}

class Main{
public static void…
   ArrayList<A> listan=new ArrayList<B>(); //FEL
   ArrayList<B> listan2=new ArrayList<B>(); //
RÄTT

}
```

10

Generiska metoder

```
//B är subklass till A
…
public void metod(ArrayList<A> list){}


ArrayList<B> listan2 = new ArrayList<B>();
ArrayList<A> listan = new ArrayList<A>();

metod(listan2) //Fel
metod(listan)  //Rätt
```

11

Binära träd

Ett träd som har max två barn kallas binärträd.

```
class Node{
      Integer value;
      Node left, right;
      public Node(Integer o){
            this.value = o;
      }
}
```

12

Binära träd

```
class SearchTree{
      private Node root;
      public SearchTree(){
          root = null;
      }
      public void put(Integer value){
            root=put(value, root);
      }
```

13

Binära träd

```
private void put(Integer value, Node pntr){
      if (pntr==null)
            pntr=new Node(value);
      else if (pntr.value > value)
            pntr=put(value,pntr.left);
      else if (pntr.value < value)
            put(value,pntr.right);
      else
            system.out.printn(”dubbla objekt”)
      return pntr;
```

14

Binära träd

```
    public void writeTree(){
         writeTree(root);
    }
    private void writeTree(Node pointer){
        if(pointer==null)
             return;
        else{
            System.out.println(pointer.value);
            writeTree(pointer.left);
            writeTree(pointer.right);
        }
    }
```

15