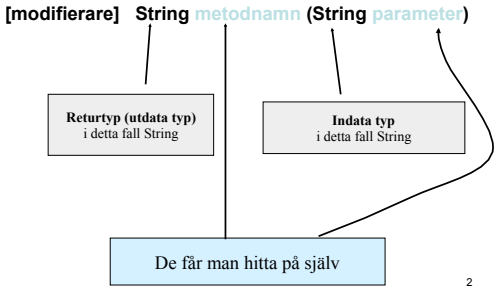
Föreläsning 2
Objektorienterad programmering DD1332
Array

1

Deklaration av metoder

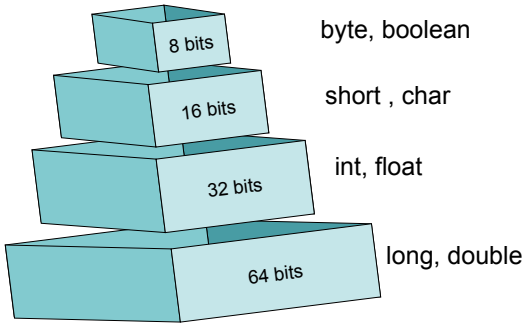


2

byte $-2^8 \ldots 2^8-1$
short $-2^{15} \ldots 2^{15}-1$
int $-2^{31} \ldots 2^{31}-1$
long $-2^{63} \ldots 2^{63}-1$
float $-2^{31} \ldots 2^{31}-1$
double $-2^{63} \ldots 2^{63}-1$

3

Typomvandling



Typ konvertering (cast)

- Används när man explicit vill ändra typen av ett uttryck.
- Exempel:

```
double x = 2.75;
int y = (int) x;
double z = y;
```

- Fungerar bara för rimliga konverteringar och i princip inte mellan primitiva och referensdatatyper.

5

Typomvandling mellan primitiva och referensdatatyper

- För att omvandla primitiva datatyper till deras motsvarande wrapper klasser är enkel t.ex:
  int i=3;
  Integer j=i;
- Men vill man till exempel omvandla en String till en int får man läsa API:n.

Typomvandling mellan referensdatatyper

- För att omvandla primitiva datatyper till deras motsvarande wrapper klasser är enkel t.ex:
  int i=3;
  Integer j=i;
- Men vill man till exempel omvandla en `String` till en `int` fårman läsa API:n.

Exempel

Skriv ett program som frågar efter radien för en cirkel och beräknar omkretsen och arean för cirkeln med hjälp av två metoder.

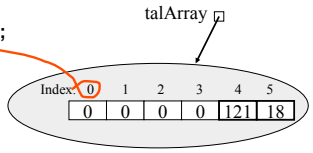
-utöka programmet sedan så att det frågar om man vill ange en ny radie varje gång, så att man kan beräkna flera gånger utan att programmet avslutas.

8

Array

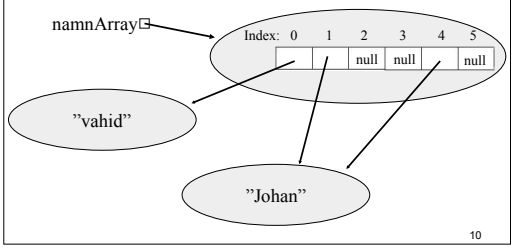
Array är en indexerad lista av element som har **samma** typ.
Man kan både ha arrayer av primitiva typer (int, double, char,...) och arrayer av referens datatyper (String, …).

```
int[] talArray=new int[6];
int t= talArray[0];
talArray[4]= 121;
talArray[5]= 18;
```



9

```
String[] namnArray=new String[6];
namnArray[0] =”vahid”;
namnArray[1]= ”Johan”;
namnArray[4]= namnArray[1];
```



10

Storlek på en array

Varje array har ett attribut som är av typen `int` och visar antal element i arrayen

```
String[] kursBeteckningar = new String[3];
int storlek = kursBeteckningar.length;
```

OBS! blanda inte ***attributen*** **length** och ***metoden*** **length()** som finns i klassen **String**

11

Metoden main och parametern args

```
public static void main (String[] args){
…
}
```

Parametern `args` **tilldelas automatisk av de värden som man anger när man kör ett java program:**
`java prg arg1 arg2 arg3…`

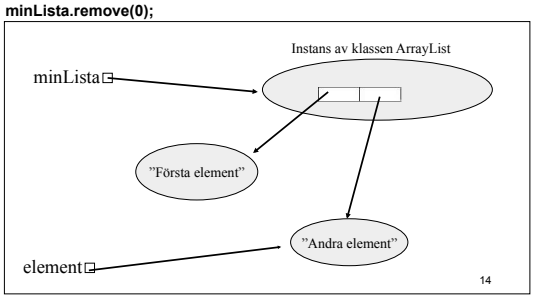
12

Klassen ArrayList

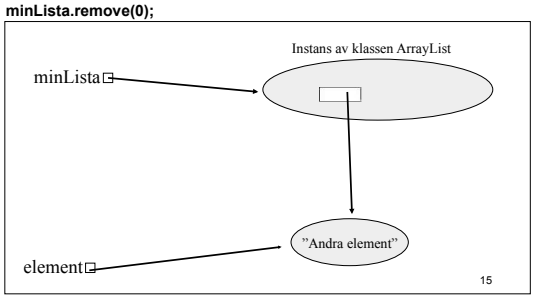
- Klassen **ArrayList** finns i paketet **java.util**, därför detta paket ska importeras om man vill använda sig av klassen **ArrayList**.
- Klassen **ArrayList** har bl.a tre viktiga ***metoder***.

```
ArrayList minLista= new ArrayList();
minLista.size(); //returnerar talet 0
minLista.add("element 1 i listan");
minLista.size(); //returnerar talet 1
minLista.add("element 2 i listan");
String e = (String) minLista.get(0);
```


ArrayList minLista= new ArrayList();
minLista.add("Första element");
minLista.add("Andra element");
String element=(String)minVek.get(1);
minLista.remove(0);



ArrayList minLista= new ArrayList();
minLista.add("Första element");
minLista.add("Andra element");
String element=(String)minVek.get(1);
minLista.remove(0);

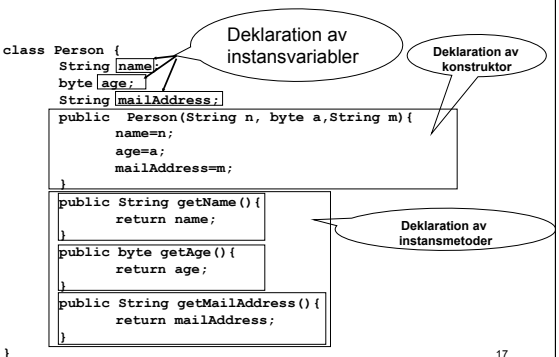


Klass och instans (objekt)

- En klass är en mall för ett objekt
  t.ex. bil, konto
- Ett objekt är en instans av en klass
  t.ex. den röda bilen, mitt lönekonto

- att skapa objekt i java:
  - kännetecken (oftast): new
    t.ex. `Scanner stdin = new Scanner(....`
  - undantag: i java API den enda klass som inte behöver
    "new" för att skapa ett objekt är klassen "String" och wrapper klasser.
    t.ex. `String namn = "Vahid"`
    `String namn = new String("Vahid");`

Instansvariabler och instansmetoder




```
class Person {
        String name;
        byte age;
        String mailAddress;
        public  Person(String n, byte a,String m){
                name=n;
                age=a;
                mailAddress=m;
        }
        public String getName(){
                return name;
        }
        public byte getAge(){
                return age;
        }
        public String getMailAddress(){
                return mailAddress;
        }
}
```


Referenser

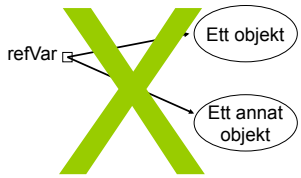
Alla variabler som används för att komma åt ett objekt är referenser.
En referensvariabel innehåller inga värde däremot adressen till någon minnesutrymme där instansen befinner sig.

exempel:
String str =" hej"; //str är en referens variabel som pekar på en instans av String.
men
int tal = 2; // tal är en variabel som har värdet 2.

- En referens kan endast referera (peka) till en instans åt gången alltså inte mer än en instans.

- En instans kan refereras (pekas) av många referenser (pekare).

Följande kan alltså aldrig hända



Programkodens uppbyggnad

Deklaration av klass {

- Deklaration av variabel1 (klassvariabler)
- Deklaration av variabel2 (instansvariabler)
- Flera ***instansvariabler*** eller ***klassvariabler*** kan deklareras här …
- Deklaration av metode1 { deklaration av ***lokala*** variabler ... }
- Deklaration av metode2 { deklaration av ***lokala*** variabler ... }
- Flera ***instansmetoder*** , ***klassmetoder*** och ***konstruktorer*** kan deklareras här …

}

Variabler

En variabel kan vara någon av följande:

- Klassvariabel, initieras automatiskt
- Instansvariabel, initieras automatiskt
- Lokalvariabel, måste initieras i koden

Kännetecken för variabler

- Kännetecken för en **klassvariabel** är:
  1. deklareras med nyckelordet **static**
  2. deklareras alltid **utanför** alla metoder men inom klassen
- Kännetecken för en **instansvariabel** är:
  1. deklareras **INTE** med nyckelordet **static**
  2. deklareras alltid **utanför** alla metoder men inom klassen
- Kännetecken för en **lokalvariabel** är:
  1. deklareras **INTE** nyckelordet **static**
  2. deklareras alltid **inom** en metod eller som formell parameter

Variabler	**deklareras**
instansvariabler	1.utanför metoder 2. utan **static**
klassvariabler	1.utanför metoder 2.med **static**
lokala variabler	1.inuti en metod 2.utan **static**

Metoder

En metod kan antingen vara en

- Klassmetod
- Instansmetod
- Konstruktor

Kännetecken för metoder

- Kännetecken för en **klassmetod** är att:
  1. deklareras med nyckelordet **static**
  2. har en returtyp
- Kännetecken för en **instansmetod** är att:
  1. deklareras **INTE** med nyckelordet **static**
  2. har en returtyp
- Kännetecken för en **konstruktor** är att:
  1. deklareras **INTE** med nyckelordet **static**
  2. har ingen returtyp och har **samma namn som klassen.**

25

Metod	deklareras
Klassmetod	• med **`static`** • med returtyp
Instansmetod	• utan **`static`** • med returtyp
Konstruktor	• utan **`static`** • utan returtyp • samma namn som klassen

26

Klassmetoder

- Normalt anropas alltid en metod via ett objekt (en instans av klassen).

```
Color röd = new Color(255,0,0);
Color mörkröd = röd.darker();
```

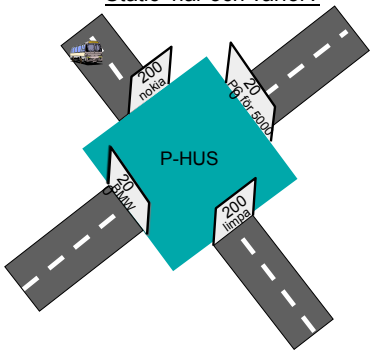
- Om en metod deklareras med static kan den anropas direkt från klassen - utan att man behöver skapa något objekt.

```
double x = Math.sqrt(17);
```

- En klassmetod kan använda sig av klassvariabler men inte instansvariabler.

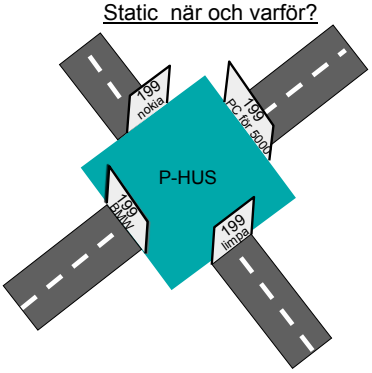
27

Static när och varför?



28

Static när och varför?



29

Filläsning

```
Scanner sc=new Scanner(newFile("valutor.txt"));
String line = null;
while(sc.hasNextLine()){
      line = sc.nextLine();
      Stystem.out.println(line );
      }
```

30