Klase i objekti u Javi

Sadržaj

- Uvod u klase, objekte i metode
- Primjer klase „Account”
- Konvencija nazivanja klasa, objekata i metoda
- Kreiranje objekta klase „Account”
- „import” deklaracije
- Primitivni i referentni tipovi
- Pisanje konstruktora
- Novi „Date and Time API” u Javi 8
- Primjer korištenja klase „LocalDateTime”
- Objekti i reference
- Uspoređivanje objekata

Uvod u klase, objekte i metode

- Uvođenjem klasa definiraju se vlastiti tipovi podataka koji se mogu višestruko iskorištavati nasljeđivanjem
- Klasama se opisuju **entiteti** sustava koji imaju svoje **atribute** (varijable), a **metodama** (funkcijama) se opisuje njihovo **ponašanje**
- Na primjer, ako se želi kreirati klasa koja predstavlja bankovni račun, može se nazvati „Account” koji ima svoje atribute „naziv” i „stanje” i sadrži metode koje omogućavaju dohvat stanja računa, te uvećavanje i smanjivanje
- Metode za dohvaćanje vrijednosti atributa nazivaju se „get” i „set” metode (mogu se automatski generirati unutar Eclipsea)

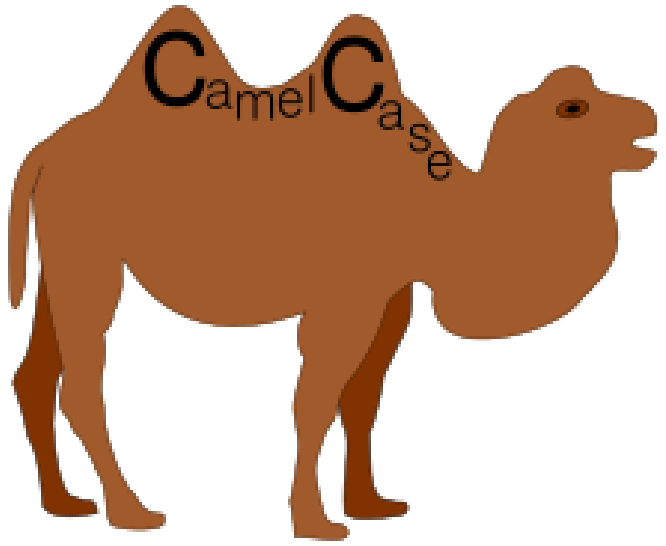
Primjer klase: „Account”

```
package account;

public class Account {

	private String name;

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}
```

- Varijabla „name” predstavlja varijablu instance, jer svaki račun (instanca/objekt klase „Account”) ima vlastiti naziv računa
- Tip varijable „name” je „String” koji predstavlja niz znakova
- Vrijednost varijable „name” ne mijenja se „izravno”, već korištenjem metode „setName”, a dohvaća se pomoću metode „getName”
- Elementi označeni modifikatorom „private” mogu se koristiti samo unutar klase, a oni označeni s „public” mogu se koristiti i izvan klase
- Parametar metode „setName” je lokalni parametar i može se koristiti samo unutar metode
- Za razlikovanje parametra i istoimene varijable instance koristi se modifikator ključna riječ „this”

Konvencija nazivanja klasa, objekata i metoda

- **Nazivi klasa** moraju započeti **velikim slovom**
- Svaka klasa mora biti spremljena u **istoimenu datoteku s ekstenzijom „.java”**
- **Nazivi objekata** moraju započeti **malim slovom**
- Izbjegavati hrvatske dijakritičke znakove
- Koristi se „CamelCase” princip radi bolje čitljivosti
- Npr.
  - „PrvaLaboratorijskaVjezba”
  - „brojacStudenata”
- **Nazivi metoda** moraju započeti **malim slovom**



Kreiranje objekta klase „Account”

```
public class AccountTest {

 public static void main(String[] args) {
   Scanner unos = new Scanner(System.in);

   Account mojRacun = new Account();

   System.out.println("Unesite naziv računa: ");
   String brojRacuna = unos.nextLine();

   mojRacun.setName(brojRacuna);

   unos.close();
 }
}
```

- Pomoću objekta klase „Scanner” moguće je dohvaćati podatke koje unosi korisnik iz konzole
- Objekt klase „Account” kreira se u zasebnoj klasi koja ima „main” metodu
- Objekte je moguće kreirati pozivom konstruktora (metode istog imena kao i klasa koja služi za kreiranje objekata)
- Konstruktor se poziva ključnom riječju „new”
- Ako se ne napiše konstruktor unutar klase, kompajler generira *defaultni* konstruktor (koji ne prima nikakve parametre i ne inicijalizira varijable)

„import” deklaracije

- Ako se unutar neke klase koriste druge klase koje nisu unutar istog paketa, potrebno ih je dodati u „import” deklaracije
- One se nalaze nakon deklaracije paketa, a prije deklaracije klase, npr.

```
package account;

import java.util.Scanner;

public class AccountTest {…
```

- Najčešće se dodaju automatski korištenjem „auto complete” kombinacije tipke („CTRL+Space”) ili korištenjem opcije „Source->Organize Imports”
- Ako se ne koristi „import”, potrebno je koristiti puni naziv klase kod programiranja:

java.util.Scanner unos = **new java.util.Scanner(System.*in*);**

Primitivni i referentni tipovi

- Java dijeli tipove na **primitivne** i **referentne**
- Primitivni tipovi su naslijeđeni iz C++-a i to su redom:
  - int, boolean, byte, char, short, long, float i double
- Referentni tipovi podataka su svi oni tipovi koji imaju vlastitu klasu
- Svaki primitivni tip ima svoju referentnu „verziju”
- Varijable primitivnih tipova mogu sadržavati samo jednu varijablu i automatski im se dodjeljuje početna vrijednost („0” ili „false”)
- Ako se referentni tipovi ne inicijaliziraju, poprimaju vrijednost „null” (ako se nad takvim objektima pozivaju metode ili dohvaćaju varijable, dogodit će se „NullPointerException”)

Pisanje konstruktora

- Ako se napiše vlastiti konstruktor, objekte klase je potrebno kreirati korištenjem tog konstruktora (*defaultni* više nije moguće koristiti)
- Konstruktor se mora isto nazivati kao i sama klasa te nema povratni tip, npr.

```
public Account(String name) {
        this.name = name;
}
```

- U tom slučaju se kod poziva konstruktora mora predati parametar koji se inicijalizira
- Osim „ručnog” pisanja konstruktora, moguće je koristiti i automatsko generiranje pomoću opcije „Source->Generate Constructor using Fields…”

Novi „Date and Time API” u Javi 8

- Do Jave 7 koristile su se Date i Calendar klase koje su bile zastarjele i neintuitivne
- S vremenom su programeri napisali vlastiti *open source library* pod nazivom JodaTime koji je znatno unaprijedio korištenje datuma i vremena u Javi
- Java 8 uvodi novi „Date and Time API” po uzoru na JodaTime
- Moguće je koristiti klase samo za datum, samo za vrijeme, oboje, dan u tjednu itd.
- Klasa „Instant” služi za korištenje trenutnog vremena i korištenja nanosekundi od početka mjerenja 01.01.1970. godine
- Klasa „LocalDate” omogućava dohvat datuma koji uključuje godinu, mjesec i dan
- Klasa „LocalTime” omogućava rad s vremenom, a „LocalDateTime” s vremenom i datumom

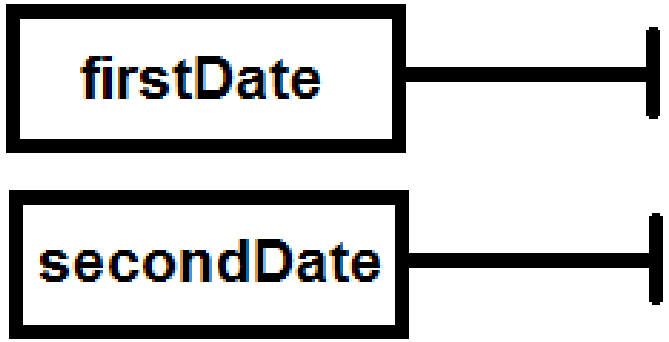
Primjer korištenja klase „LocalDateTime”

```
LocalDateTime localDateTime = LocalDateTime.now();
System.out.println("Trenutni datum: " + localDateTime);

System.out.println("Nakon dva tjedna: " + localDateTime.plusWeeks(2));

System.out.println("Formatirani datum: " + localDateTime.format(
        DateTimeFormatter.ofPattern("dd.MM.yyyy.")));
```

```
Trenutni datum: 2014-10-10T12:37:29.035
Nakon dva tjedna: 2014-10-24T12:37:29.035
Formatirani datum: 10.10.2014.
```

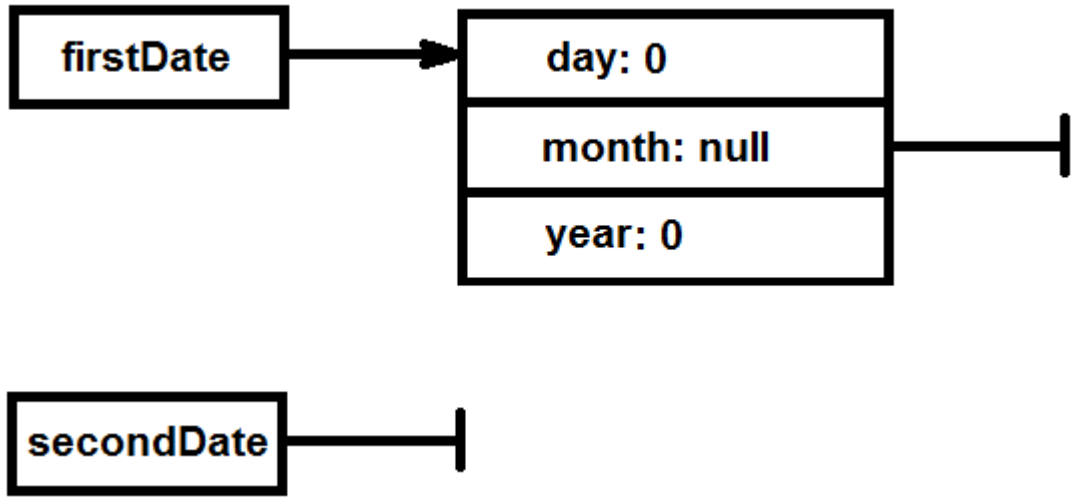
Objekti i reference (1)

- Nakon što se objekti deklariraju i ne dodijeli im se konkretna vrijednost (referenca), imaju vrijednost „null”, što se može prikazati na sljedeći način (npr. Objekti „firstDate” i „secondDate”):



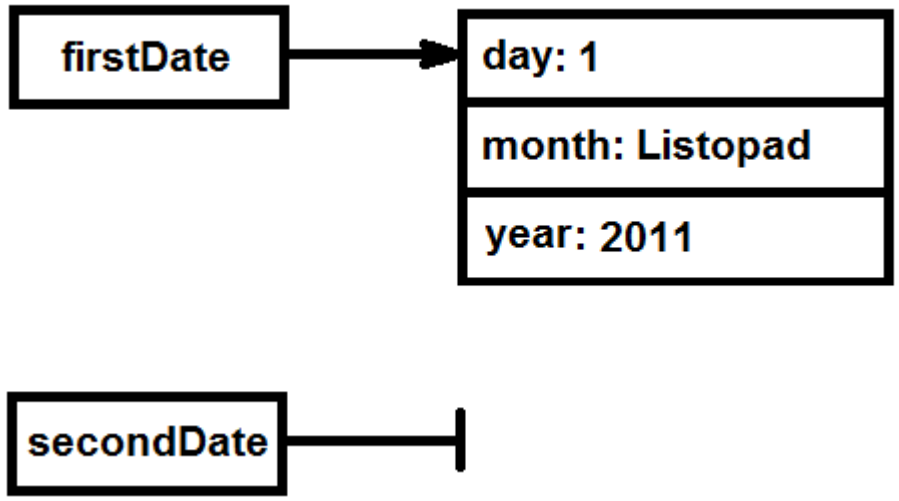
Objekti i reference (2)

- Kreiranjem objekta „firstDate”, npr. Korištenjem *defaultnog* konstruktora, rezervira se potrebna memorija za spremanje podataka
- Primitivni tipovi (npr. „int”) poprimaju vrijednost „0”, a referentni (npr. „String” poprimaju vrijednost „null”):



Objekti i reference (3)

- Dodjeljivanjem konkretnih vrijednosti za članske varijable (npr. pomoću „setter” metoda) moguće je promijeniti inicijalne vrijednosti:

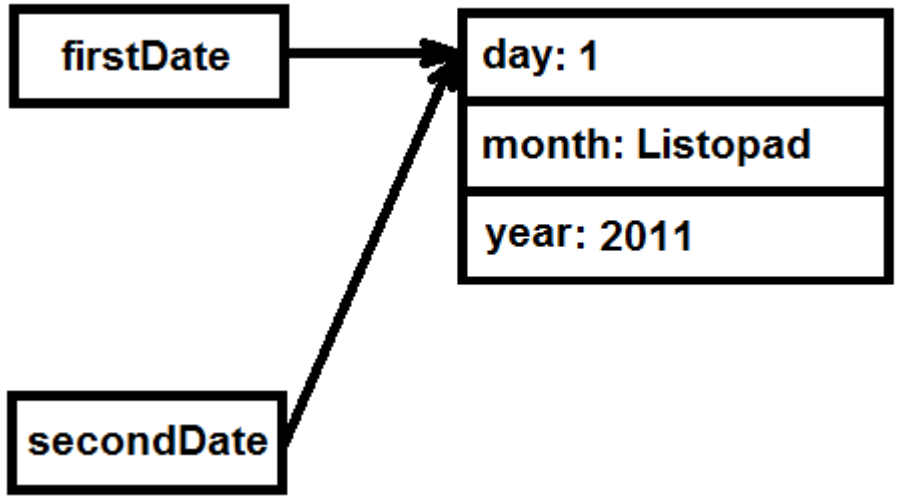


Objekti i reference (4)

- Naredbom:

  secondDate = firstDate;

- objekti počinju dijeliti istu referencu, a samim time i iste memorijske lokacije s podacima (promjenom podataka će oba objekta imati novu vrijednost):



Uspoređivanje objekata

- Objekti se u Javi mogu uspoređivati na dva načina:
- Po referencama (često se koristi pogrešno):
  - **if(firstDate == secondDate) {…}**
- Po vrijednostima, za što je najbolje napisati svoju metodu (ili nadjačati metodu „equals”):

```
boolean areTwoDatesEqual(Date f, Date s) {
  if (f.day == s.day && f.month.equals(s.month) && f.year == s.year)
    return true;
  else
    return false;
}
```

Pitanja?