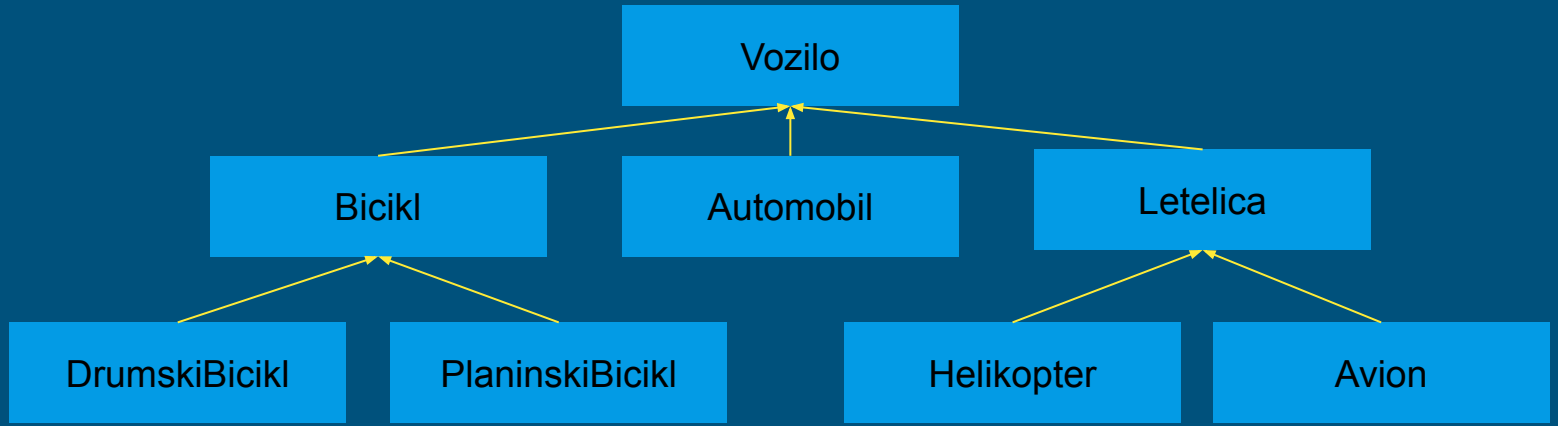
7. Proširivanje

Sadržaj

- Proširivanje
- Klasa `Object`
- Redefinisanje metoda
- Inicijalizacija objekta izvedene klase

Pojam proširivanja

- Implementacija OO koncepta specijalizacije
- Predstavlja odnos "*is a kind of*" između izvedene i osnovne klase



Terminologija i pojmovi

- Specijalizacija
- Generalizacija
- Osnovna klasa
- Izvedena klasa
- Proširivanje
  - izvedena klasa proširuje osnovnu klasu
- Nasleđivanje
  - posledica proširivanja
  - izvedena klasa nasleđuje obeležja i metode osnovne klase

Proširivanje

- Izvedena klasa nasleđuje obeležja i metode osnovne klase
- Izvedena klasa proširuje osnovnu klasu tako što definiše dodatna obeležja i metode
- Izvedena klasa može da redefiniše nasleđene operacije, odnosno da ima drugačije ponašanje od nasleđenog
- Prilikom deklaracije izvedene klase koristi se rezervisana reč `extends` i navodi naziv osnovne klase
- Java (i većina drugih OO jezika) ne podržava višestruko proširivanje - može se specificirati samo jedna osnovna klasa

Definicija klase `Donut`

- Krug sa rupom je specijalna vrsta kruga koja pored centra i radijusa ima i unutrašnji radijus (radijus rupe)

```
public class Donut extends Circle {
  private int innerRadius;
  // konstruktori

  public int getInnerRadius() {
    return this.innerRadius;
  }
  public void setInnerRadius(int innerRadius) {
    this.innerRadius = innerRadius;
  }
}
```

Pristup nasleđenim obeležjima i metodama

- Privatnim obeležjima i metodama nije moguće pristupiti direktno van njihove klase, pa time ni iz izvedene klase
- Specifikator pristupa `protected` omogućava pristup obeležjima i metodama neke klase u njenim izvedenim klasama
- Ukoliko se obeležja deklarišu kao `private`, u izvedenim klasama moguće im je pristupiti samo putem metoda pristupa

Redefinisanje metode u izvedenoj klasi

- Kada nasleđena metoda ne odgovara ponašanju izvedene klase, potrebno je izvršiti njeno redefinisanje
- Metoda se redefiniše deklaracijom nove metode u izvedenoj klasi koja ima isti potpis, odnosno:
  - modifikator pristupa,
  - povratni tip podatka,
  - naziv i
  - listu parametara
- Redefinisanu metodu osnovne klase moguće je pozvati korišćenjem rezervisane reči `super`

Metoda `contains` u klasi `Donut`

- Nasleđena metoda `contains` ne odgovara klasi `Donut` jer tačke unutar rupe ne pripadaju krugu sa rupom
- Metoda `contains` treba da bude redefinisana u klasi `Donut` na sledeći način:

```
public boolean contains(Point p) {
  return (center.distance(p) <= radius) && (center.distance(p) >= innerRadius);
}
```

Metoda `area` u klasi `Donut`

- Nasleđena metoda `area` ne odgovara klasi `Donut` jer od površine kruga treba oduzeti površinu rupe
- Redefinisanu metodu `area` moguće je iskoristiti kako se ne bi ponavljao kod izračunavanja površine kruga
- Metoda `area` treba da bude redefinisana u klasi `Donut` na sledeći način:

```
public double area() {
  return super.area() - innerRadius * innerRadius * Math.PI;
}
```

Operator `instanceof`

- Omogućava proveru tipa objekta u vreme izvršenja programa

```
public void someMethod(Circle c) {
  if (c instanceof Donut) {
    Donut temp = (Donut)c;
    temp.setInnerRadius(5);
  }
}
```

- Krug sa rupom jeste krug, ali ne važi i obrnuto

Klasa `Object`

- Podrazumevana osnovna klasa za sve klase
- Metode koje se često redefinišu:
  - `public String toString()`
  - `public boolean equals(Object)`
- Metoda `toString` se redefiniše kako bi se obezbedila reprezentacija stanja objekta u obliku niza znakova
- Metoda `equals` se redefiniše kako bi se uporedilo stanje objekata na jednakost

Metoda `toString` u klasi `Point`

- Pretpostavimo da želimo stanje svakog objekta `Point` da predstavimo pomoću niza znakova u sledećem obliku:

  `(x, y)`
- U tom slučaju potrebno je redefinisati metodu `toString` na sledeći način:

```
public String toString(){
  return “(” + this.x + “,” + this.y + “)”;
}
```

Metoda `equals` u klasi `Point`

- Ukoliko je uslov jednakosti dva objekta klase `Point` da su im iste koordinate i selektovanost, metoda `equals` bi bila redefinisana na sledeći način:

```
public boolean equals(Object obj){
  if (obj instanceof Point) {
    Point temp = (Point)obj;
    if (x == temp.x && y == temp.y && selected == temp.selected)
      return true;
    else
      return false;
  } else
    return false;
}
```

Kreiranje objekta izvedene klase

- Pri pozivu konstruktora izvedene klase potrebno je proslediti sve parametre potrebne za inicijalizaciju definisanih obeležja

```
Donut d = new Donut(new Point(6,3), 8, 5);
```

- Odgovornost svake od klasa u hijerarhiji nasleđivanja je da izvrši inicijalizaciju svojih obeležja

Inicijalizacija objekta izvedene klase

- Konstruktor izvedene klase najpre poziva konstruktor osnovne klase pa potom vrši inicijlizaciju svojih obeležja
- Poziv konstruktora osnovne klase vrši se pomoću rezervisane reči `super`

```
public Donut(Point p, int radius, int innerRadius) {
  super(p, radius);
  this.innerRadius = innerRadius;
}
```

Poziv konstruktora osnovne klase

- Konstruktor izvedene klase najpre poziva konstruktor osnovne klase pa potom vrši inicijlizaciju svojih obeležja
- Ukoliko u konstruktoru izvedene klase nije eksplicitno pozvan konstruktor osnovne klase, implicitno se poziva konstruktor bez argumenata
- Ukoliko u izvedenoj klasi nije definisan konstruktor, kompajler generiše podrazumevani konstruktor koji poziva konstruktor bez parametara u osnovnoj klasi

Rezervisana reč `final`

- Primitivni tip - vrednost ne može da se menja

```
final int x = 5;
```

- Referenca na objekat - referenca ne može da se menja

```
final Point p = new Point(6, 8);
```

- Sprečavanje proširivanja

```
public final class MyColor {...}
```

- Sprečavanje redefinisanja metode

```
public class … {
  public final boolean passwordOK(String pass) {...}
}
```

Rezime

- Proširivanje je implementacija OO koncepta specijalizacije
- Proširivanje omogućava višestruko korišćenje koda
- Klasa Object je podrazumevana osnovna klasa za sve definisane klase
- Redefinisanje nasleđene metode omogućava promenu ponašanja objekta izvedene klase u odnosu na objekte osnovne klase
- Inicijalizacija objekta izvedene klase izvršava se tako što se inicijalizuju obeležja od gore prema dole u hijerarhiji prioširivanja