10. Izuzeci

Sadržaj

- Nepredviđene situacije u vreme izvršenja programa
- Obrada izuzetaka
- Definisanje novog tipa izuzetka

Pojam izuzetka

- Izuzetak (engl. *exception*) je događaj koji predstavlja pojavu greške i promenu normalnog izvršenja programa
- Java omogućava obradu izuzetaka na uniforman način
- Omogućava da se kod za obradu izuzetaka odvoji od glavne logike programa

Pojava izuzetka

- Pojava greške u vreme izvršenja programa ima za posledicu bacanje izuzetka

  `throw <exceptionObject>;`
- Ukoliko nije uhvaćen izuzetak, doći će do prekida izvršenja programa i ispisa poruke o grešci na konzolu

```
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
        at security.AES.toStringBuilder(AES.java:165)
        at security.AES.encrypt(AES.java:91)
        at network.Client.send(Client.java:63)
        at network.Manager.ask(Manager.java:50)
        at gui.GUI.buttonFilmActionPerformed(GUI.java:579)
        at gui.GUI.access$100(GUI.java:15)
        at gui.GUI$2.actionPerformed(GUI.java:113)
        at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2018)
        at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2341)
        at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:402)
        at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:259)
        at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:252)
        at java.awt.Component.processMouseEvent(Component.java:6505)
        at javax.swing.JComponent.processMouseEvent(JComponent.java:3321)
        at java.awt.Component.processEvent(Component.java:6270)
        at java.awt.Container.processEvent(Container.java:2229)
        at java.awt.Component.dispatchEventImpl(Component.java:4861)
        at java.awt.Container.dispatchEventImpl(Container.java:2287)
        at java.awt.Component.dispatchEvent(Component.java:4687)
```

Primer bacanja i prosleđivanja izuzetka

```
public class Circle extends Shape {
  // …

  public void setRadius(int radius) throws Exception {
    if (radius <= 0)
      throw new Exception("Poluprecnik mora biti > 0.");
    this.radius = radius;
  }
}
```

Enkapsulacija i modifikatori pristupa

- Javni pristup obeležju `radius` bi omogućio njegovo postavljanje na negativnu vrednost

```
Circe c = new Circle(...);
c.radius = -5;
```

- Privatni pristup obeležju i `setRadius` metoda nam omogućavaju da sprečimo postavljanje obeležja `radius` na negativnu vrednost
- Izuzeci nam omogućavaju da detektovanu nepredviđenu situaciju u vreme izvršenje programa obradimo i reagujemo u skladu sa potrebama
  - prikaz dijaloga sa porukom
  - ispisom poruke na konzolu
  - ...

Hvatanje izuzetaka

- Pozivi metoda koje bacaju izuzetak smeštaju se u `try` blok
- U `catch` blokovima moguća je obrada različitih tipova izuzetaka

```
Circle c = new Circle(...);
String strRadius = txtRadius.getText();
try {
  int radius = Integer.parseInt(strRadius);
  c.setRadius(radius);
} catch(NumberFormatException nfe) {
  System.out.println("Nije upisana celobrojna vrednost. " +  nfe.getMessage());
} catch(Exception e) {
  System.out.println("Greska: " +  e.getMessage());
}
```

Blok `finally`

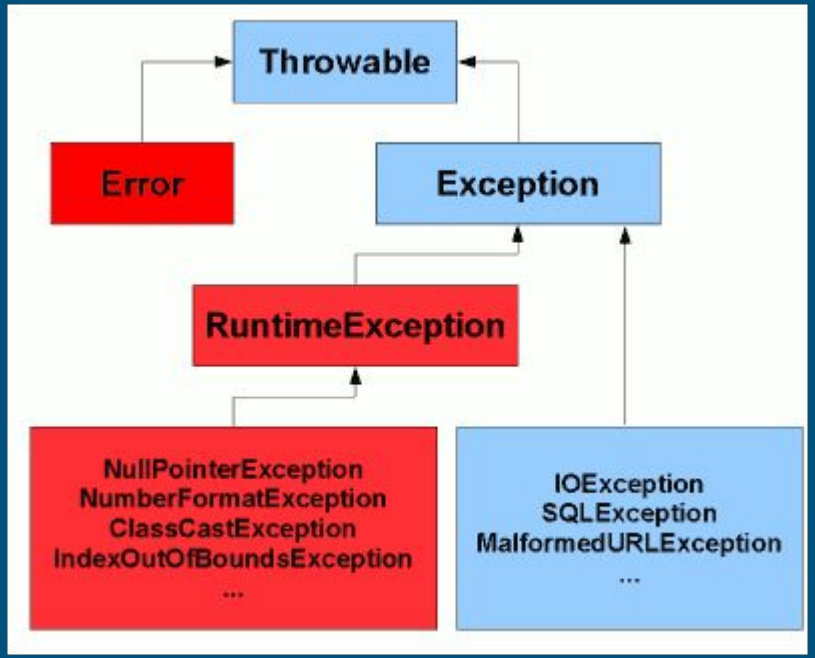
- Ukoliko se zauzmu neki resursi i potom desi izuzetak, potrebno ih je osloboditi
- Kod za oslobađanje resursa se smešta u `finally` blok
- `finally` blok se izvršava čim se izađe iz `try` bloka (sa ili bez pojave izuzetka)

```
try {
  // kod koji moze da izazove pojavu izuzetka
} catch(Exception ex) {
  // kod za obradu izuzetka
} finally {
  // “cleanup” kod
}
```

Pristupi upravljanju izuzecima

- Pokušati sprečiti pojavu izuzetka
  - ograničenja u korisničkom interfejsu
- Uhvatiti izuzetak u metodi u kojoj se pojavljuje
- Deklarisati da metoda baca izuzetak i proslediti ga
- Ukoliko se ne uhvati izuzetak, program će završiti sa izvršenjem i ispisati *stack trace*
- Ispis *stack trace*-a je korisno u vreme razvoja i može se izvršiti pozivom metode `printStackTrace` koja je definisana u klasi `Exception`

Hijerarhija klasa izuzetaka



Java kompajler i provera izuzetaka

- Java kompajler proverava da li su svi izuzeci **koji se proveravaju**
  - uhvaćeni ili
  - dalje prosleđeni (bačeni)
- Ne proveravaju se
  - `Error` i sve izvedene klase
  - `RuntimeException` i sve izvedene klase
- Na primer, proveravanje `NullPoinerException` izuzetka bi zahtevalo pisanje `try - catch` blokova gde god se pojavljuje poziv metode za referencu na objekat

Klase `Error` i `RuntimeException`

- `Error` klasa predstavlja situacije od kojih program ne može da se oporavi
  - `OutOfMemoryError`
  - `StackOverflowError` …
- `RuntimeException` je osnovna klasa za izuzetke koje nema smisla proveravati
  - `NullPointerException`
  - `ClassCastException`
  - `ArrayIndexOutOfBoundsException` …

Definisanje novog tipa izuzetka

- Ukoliko osim poruke i mesta pojave izuzetka želimo da objekat izuzetka sadrži i dodatne informacije, možemo definisati novu klasu izuzetka

```
public class CircleException extends Exception {
  private Circle circle;

  public CircleException(String message, Circle circle) {
    super(message);
    this.circle = circle;
  }

  public Circle getCircle() {
    return this.circle;
  }
}
```

Rezime

- Izuzeci omogućavaju obradu nepredviđenih događaja u vreme izvršenja programa
- Java kompajler kontroliše da li su svi izuzeci koji se proveravaju obrađeni
- Moguće je definisati nove tipove izuzetaka koji će sadržati dodatne informacije o nepredviđenom događaju