Objektno-orijentisano programiranje

Interfejs Cloneable i izuzeci

Nevena Ćirić
www.matf.bg.ac.rs/~nevena_ciric

Metod clone()

- Svaka klasa nasleđuje od klase Object metod clone() koji se može koristiti za pravljenje kopija (klonova) objekata.* Potpis metoda je:

  protected Object clone() throws CloneNotSupportedException;

- Metod je implementiran tako da vrši tzv. memcopy – kreira novi objekat koji je istog memorijskog sadržaja (samim tim i istog tipa) kao što je objekat nad kojim je metod pozvan.
- To znači da se vrednost svakog od atributa novog objekta postavlja na istu vrednost koju ima odgovarajući atribut tekućeg objekta, tj. da metod vrši plitko kopiranje.
- Kada objekat ima kao atribute reference ka mutirajućim objektima ovakva realizacija kloniranja u većini slučajeva nije zadovoljavajuća i treba je predefinisati.

* clone() metod je efikasnija alternativa za copy konstruktor!

Metod clone()

- Metod je deklarisan kao protected, pa mu se može pristupiti samo iz paketa java.lang u kome je definisana klasa Object ili iz potklasa klase Object preko reference super.
- Drugim rečima, metod clone() iz klase Object može da se poziva sa:
  - x.clone() u okviru klasa iz paketa java.lang
  - super.clone() u okviru definicije bilo koje klase*
- Da bismo mogli da pozivamo metod sa x.clone() u okviru klase koja nije iz paketa java.lang (npr. u main() metodu našeg programa) neophodno je predefinisati ga u klasi X sa odgovarajućim modifikatorom vidljivosti.
- To znači da čak i u slučaju kada nam odgovara "plitka" implementacija nasleđenog metoda clone() , kako bismo mogli da ga pozivamo za objekte neke klase van njene definicije potrebno je predefinisati ga pozivom super.clone()** i odgovarajućim modifikatorom vidljivosti.

* zato što su sve klase potklase klase Object

** lanac pozivanja super.clone() se nastavlja do Object.clone() koji vrši memcopy

Interfejs Cloneable

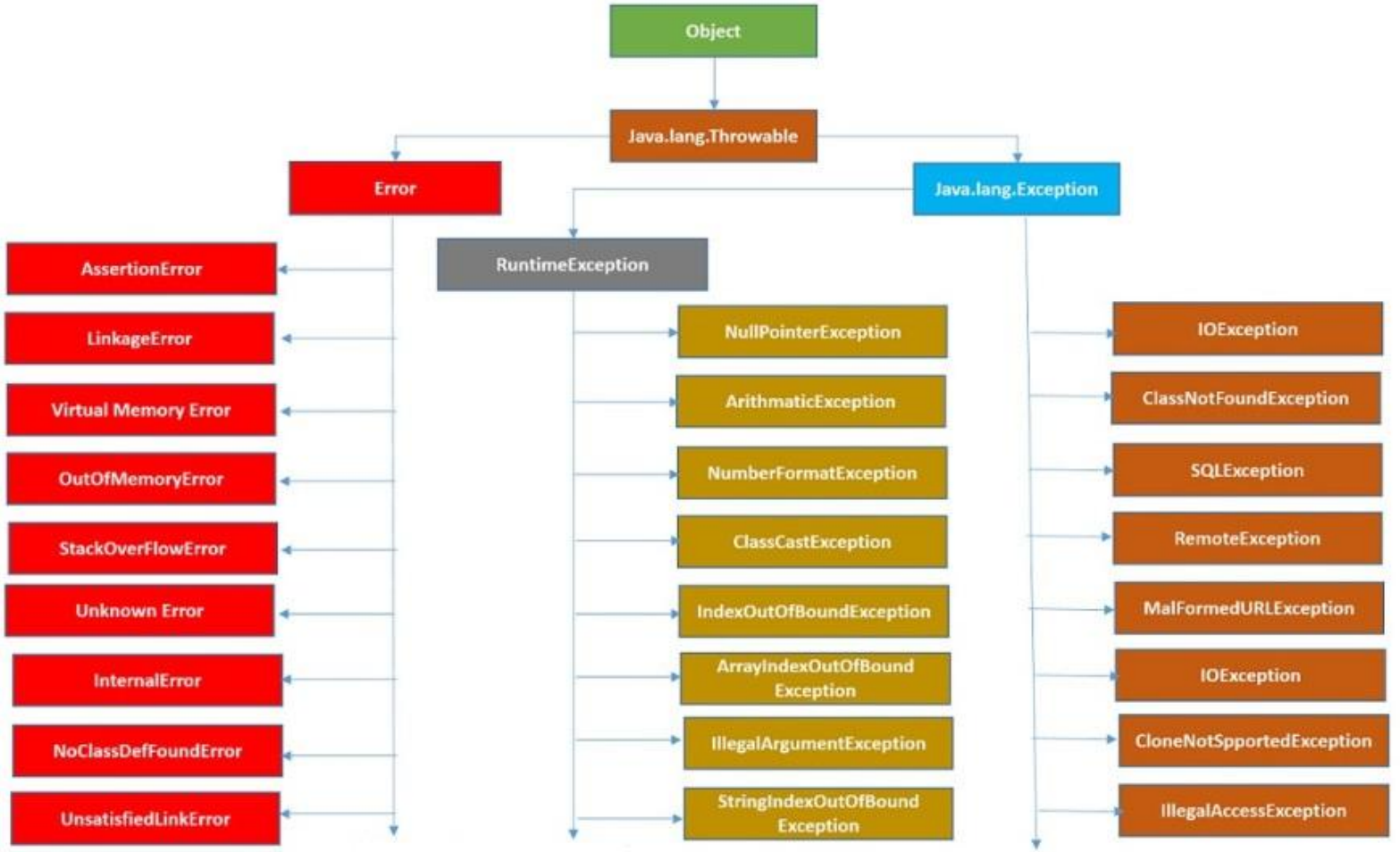
- Metod je implementiran tako da generiše izuzetak ukoliko se pokuša poziv metoda nad objektom klase koji ne implementira interfejs Cloneable.*
- Definicija Cloneable interfejsa:
  public interface Cloneable{ }
- Interfejsi koji nemaju deklarisan nijedan (apstraktan) metod, tj. ne zahtevaju implementaciju neke dodatne funkcionalnosti nazivaju se marker interfejsi.
- Neki od marker interfejsa definisanih u standardnoj biblioteci klasa su: Serializable, Clonnable, Remote.
- Upotreba interfejsa Cloneable nema nikakve veze sa uobičajenom upotrebom interfejsa.* U ovom slučaju interfejs služi kao marker koji ukazuje da dizajner klase razume proces kloniranja i želi da ga podrži.

* metod clone() nije član interfejsa Cloneable!

Izuzeci

- Mehanizmom izuzetaka vrši se signalizacija problema pri izvršavanju programa.
- Izuzetak u Javi je objekat koji sadrži informacije o prirodi problema i stanju izvršavanja programa kada je problem nastao.
- Ovi objekti se kreiraju kada dođe do greške u programu ili neuobičajenih događaja koji zahtevaju posebnu obradu.
- Kada se kreira izuzetak zaustavlja se izvršavanje programa i izuzetak se prosleđuju JRE, koji zatim pretražuje stek poziva za mesto (metod) u programu na kome će izuzetak biti obrađen i odakle će se nastaviti sa izvršavanjem programa. U suprotnom, program se završava sa odgovarajućom porukom o grešci.
- Za prosleđivanje izuzetaka JRE-u koristi se ključna reč throw. Zato se ovaj mehanizam još naziva “bacanje i hvatanje izuzetaka”.

Hijerarhija klasa izuzetaka



Hijerarhija klasa izuzetaka

- Svaki izuzetak je objekat neke potklase standardne klase Throwable.
- Klase izuzetaka su hijerarhijski organizovane tako da grupišu slične vrste izuzetaka.
- Potklase Error i Exception klase Throwable pokrivaju sve standardne izuzetke.
- Izuzeci klase Error predstavljaju ozbiljne probleme. Za njih se ne očekuje da budu obrađeni, tj. izuzeci ovog tipa se ne hvataju. Jedino što možemo da uradimo je da pročitamo poruku o grešci koja se generiše kada se izbaci izuzetak.
- Za skoro sve izuzetke klase Exception kompajler zahteva da budu obrađeni na odgovarajući način. Grupa izuzetaka iz potklase RuntimeException se drugačije tretira. Kako ova vrsta izuzetaka obično nastaje usled ozbiljnijih grešaka u logici programa, u većini slučajeva se ne može se ništa uraditi po pitanju oporavka od greške, pa kompajler dozvoljava da se ovakvi izuzeci ignorišu.

Obrada izuzetaka

- Za sve ostale potklase klase Exception kompajler zahteva da:
  - metodi koji mogu da izbace izuzetak moraju da deklarišu sve tipove izuzetaka koji mogu biti izbačeni pomoću ključne reči throws
  - metodi koji pozivaju metod koji može da izbaci izutetak moraju da izvrše obradu izuzetka ili propagiranje izuzetka
- Propagiranje izuzetka podrazumeva prosleđivanje izuzetka pozivajućem metodu. To se postiže:
  - izostankom obrade izuzetka ili hvatanjem i ponovnim izbacivanjem uhvaćenog izuzetka
  - deklarisanjem metoda (koji poziva metod koji može da izbaci izuzetak) da može da izbacuje izuzetke odgovarajućeg tipa
- Obrada izuzetka podrazumeva hvatanje izuzetka i oporavak programa od greške kako bi se nastavilo izvršavanja programa.

Obrada izuzetaka

- U Javi postoje dve vrste konstrukcija za obradu (hvatanje) izuzetaka:
  - try-catch-finally blok
  - try-with-resourses blok