MEHANIZMI ZA OBRADU IZUZETAKA

Pojam izuzetka

- Izuzetak je svaki neobičan događaj (pogrešan ili ne) koji se može otkriti hardverski ili softverski, a koji zahteva posebnu obradu
- Izuzetak treba posebno obraditi izvan osnovnog toka programa.
- Primeri standardnih izuzetaka:
  - U izrarzu se pojavilo deljenje nulom,
  - Pri rezervaciji memorijskog prostora, u dinamičkoj zoni memorije nije bilo dovoljno mesta,
  - Došlo je do prekoračenja opsega pri pristupu elementima niza,
  - Eksterni uredjaj sa kojeg se čitaju podaci je u kvaru,
  - Došlo se do kraja datoteke iz koje se čitaju podaci...

Reakcija na standarnde izuzetke

- U jezicima koji ne podržavaju obradu izuzetaka:
  - kontrola se prenosi operativnom sistemu koji obično štampa odgovarajuću poruku i prekida dalje izvršavanje programa.

- U jezicima koji podržavaju obradu izuzetaka:
  - U tačkama gde je problem nastao prijavljuje se izuzetak
  - U posebnim delovima koda se izuzetak obrađuje – traži se mogućnost da se problem reši, ili se nalazi alternativni način nastavka rada programa.

Tehnike za obradu izuzetaka

- Kod novijih programskih jezika razradjeni su mehanizmi za
  - Prihvatanje standardnih hardverskih izuzetaka i
  - Definisanje i obradu softverskih izuzetaka.

- Kod starijih programskih jezika ovi su se problemi obično rešavali pomoću
  - Korišćenja statusnih parametara ili povratne vrednosti funkcije za detkciju izuzetaka
  - Dodavanje kao parametra potprogramu labele na koju se izvršenje nastavlja ukoliko u potprogramu dođe do greške
  - Prenos kroz listu parametara potprograma koji obrađuju izuzetke

Korišćenje statusnih parametara za detekciju izuzetaka

Tehnika zasnovana na korišćenju statusnih parametara:

- Potprogramu se dodaje ekstara parametar koji nosi informaciju o statusu određenog događaja.
- Kada nastane događaj koji se kontroliše taj parametar menja vrednost.
- Pri povratku iz potprograma uvek se ispituje vrednost statusnog parametra

Nedostaci ove tehnike:

- Kod postaje jako nepregledan (gomilaju se if naredbe za ispitivanje statusnih promenljivih);
- Propagacija izuzetka kroz veći broj pozivanih procedura zahteva ispitivanje statusnog parametra u svakoj proceduri).

Različite tačke povratka iz funkcije zavisno od toga da li se izuzetak pojavio ili ne

- To omogućava da se izlaz iz potprograma razlikuje za slučaj kada se program normalno izvršava i za slučaj kada je nastao izuzetak koji se kontroliše.

- Kod jezika sa dinamičkom ili polu-dinamičkom implementacijom ova tehnika se ne može koristiti jer otežava manipulisanje aktivacionim slogovima jer izlaska iz potprograma razlikuju u zavisnosti od situacije nastale u toku izvršavanja potprograma.

Različite tačke povratka iz funkcije zavisno od toga da li se izuzetak pojavio ili ne

- Primer:
  - U programskom jeziku FORTRAN postoji mogućnost da se u okviru READ naredbe definiše labela na koju se prenosi upravljanje kada se u toku izvršavanja ove naredbe dođe do kraja datoteke koja se čita, ili nastane neka greška pri čitanju:

READ(UNIT=5,FTM=1000,ERR=100,END=999) X

Potprogram za obradu izuzetaka koji se prosleđuje kao parametar

- Potprogram za obradu izuzetaka je odvojen modul i kao parametar prenosi u potprograme u kojima se kontroliše određeni izuzetak.

- Osnovni nedostatak kod ovakvog pristupa je u tome što se pri svakom pozivu mora prenosiiti i taj potprogram koji obrađuje izuzetak. Ako u jednom potprogramu kontrolišemo više izuzetaka kod postaje pretrpan takvim elementima.

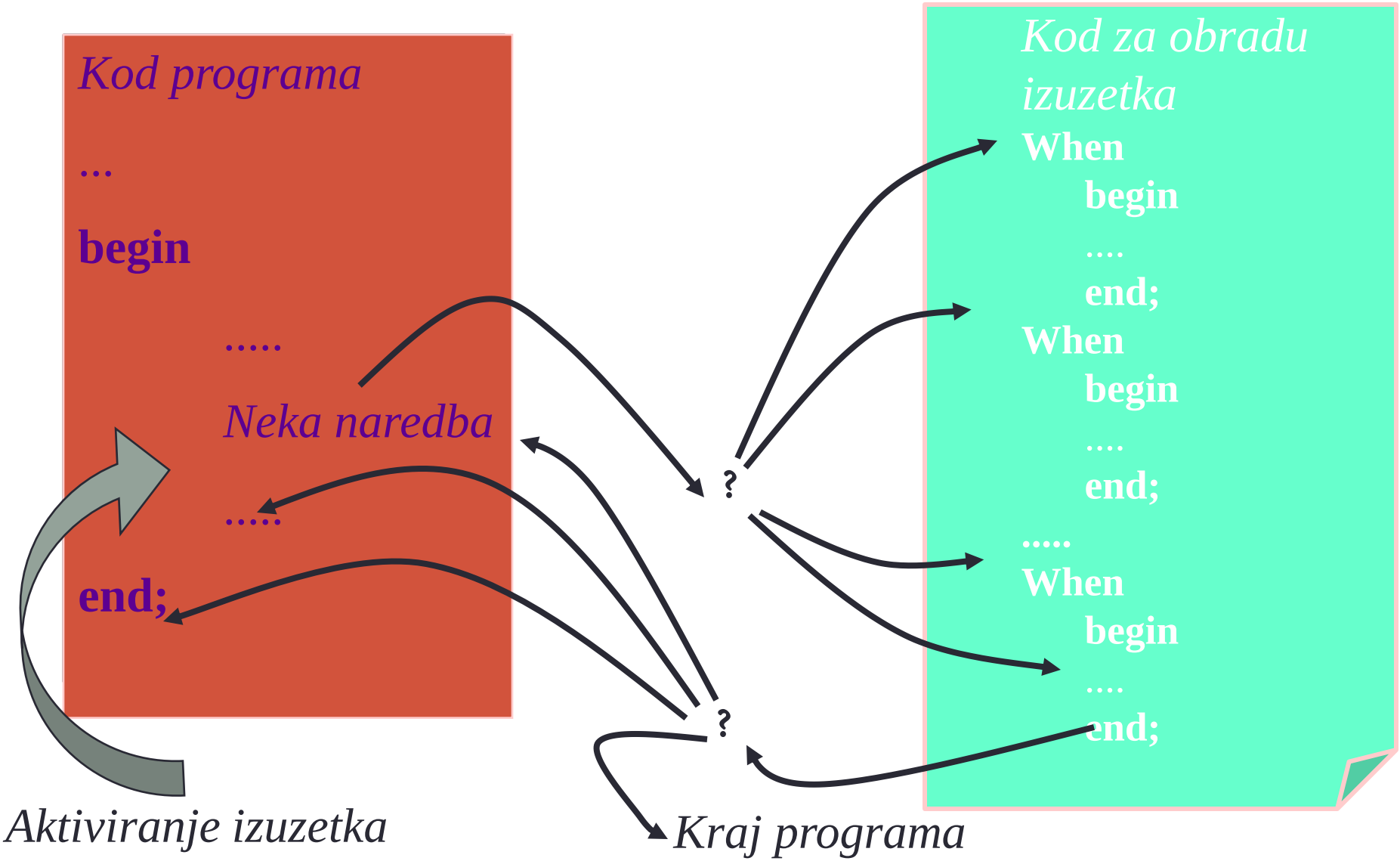
Problemi pri implementaciji obrade izuzetaka

- Gde i kako se definišu blokovi za prihvatanje i obradu izuzetaka i koji je njihov opseg?
- Kako se ostvaruje veza između prijavljenog izuzetka i bloka koji ga obrađuje (event handler-a)?
- Gde se izvršenje nastavlja kad se *event hendler* završi?
- Da li je predviđen neki oblik „finalizacije“?

Problemi pri implementaciji obrade izuzetaka

- Kako se definišu korisnički izuzeci?
- Da li treba da postoje podrazumevani event handleri ako programer ne difiniše svoje?
- Da li postoje predefinisani izuzeci?
- Mogu li da se prijave predefinisani izuzeci?
- Da li greške koje se detektuju na hardveru treba da se tretiraju kao izuzeci koji se mogu obraditi?
- Da li se i kako izuzeci mogu onemogućiti?

Tok upravljanja pri obradi izuzetaka



Izuzeci u programskom jeziku C++

Izuzeci u programskom jeziku C++

- Izuzetak u programskom jeziku C++ je privremeni objekat bilo kog tipa koji se koristi za prijavu greške
  - Može biti nekog osnovnog tipa: int, char*,...
  - Objekat neke standardne klase za predstavljanje izuzetaka
  - Objekat klase za predstavljanje izuzetaka definisane od strane programera
    - Preporuka je da takva klasa bude izvedena iz klase *exception*, ali nije obavezujuće
- Kada se problem otkrije izuzetak se prijavljuje (*exception throwing*)
- Kod koji prijavljuje izuzetak mora da stoji unutar bloka koji izuzetak obrađuje (*exception handling*)

Prijava i obrada izuzetaka u C++-u

- Prijava izuzzetaka:

```
throw Izraz;
```

- Obrada izuzetaka:

```
try {
        // Kod koji može da prijavi izuzetke
}
catch(ExceptionType1 ex) {
        // Obrada izuzetka tipa ExceptionType1
}
⋮
catch(ExceptionTypeN ex) {
        // Obrada izuzetka tipa ExceptionTypeN
}
catch(...) {
        // Obrada izuzetka proizvoljnog tipa
        // koji nije ni jedan od gornjih
}
```

Funkcijski *try-catch* blok

- Ako celo telo funkcje treba da se nađe u *try*-bloku, definiše se funkcijski *try-catch* blok:

```
int f(int k)
try
{
      // telo funkcije
}
catch(...)
{
      // Obrada izuzetka
      return x;  // Catch-blok mora da
                 // vrati rezultat
                 // odgovarajućeg tipa

}
```

Prihvatanje izuzetaka u delu za inicijalizaciju konstruktora

```
MyClass::MyClass(int k, float r)
try : SuperClass(k)
{
  // telo konstruktora
}
catch(...)
{
     // Obrada izuzetka
}
```

Funkcije koje prijavljuju izuzetke (propagiranje izuzetaka)

- Ukoliko u funkicji postoji *throw* naredba, van *try-catch* strukture koja taj tip izuzetka obradjuje, kaže se da funkcija prijavljuje izuzetak.
- U deklaraciji ili definiciji funkcije **može,** ali **ne mora**, da se navede spisak tipova izuzetaka koje funkcija prijavljuje.
- Spisak tipova izuzetaka koje funkcija prijavljuje se navodi iza zaglavlja funkcije u opisu

```
throw ( NizTipova )
```

- Ako ovakav opis stoji u definiciji funkcije, a funkcija prijavi izuzetak tipa koji nije u nizu identifikatora, kompajler će prijaviti grešku.
- Kompajler će prijaviti grešku i ako se ovakva funkcija pozove a izuzeci navedenih tipova se nigde ne obrade.
- Ako iza zaglavlja funkcije ne stoji lista tipova izuzetaka koje funkcija može da prijavi, funkcija sme da prijavi izuzetak proizvoljnog tipa.

Primer funkcije koja prijavljuje izuzetke

```
void radi(...) throw(char*, int)
{
 if(...)
      throw "Izuzetak!";
 if(...)
      throw 100;
 if(...)
      throw Tacka(0,0);
      // GRESKA: nije dozvoljeno da funkcija
      // prijavi izuzetak tipa Tacka
}
```

Funkcije koje ne prijavljuju izuzetke

- Funkciji se može „zabraniti“ da prijavljuje izuzetke na 2 načina:
  - Navođenjem prazne liste tipova u throw klazzuli iza zaglavlja:

```
void radi(...) throw();
```

  - Navođenjem *noexcept* ključne reči iza zaglavlja funkcije (od verzije C++11):

```
void radi(...) noexcept;
```

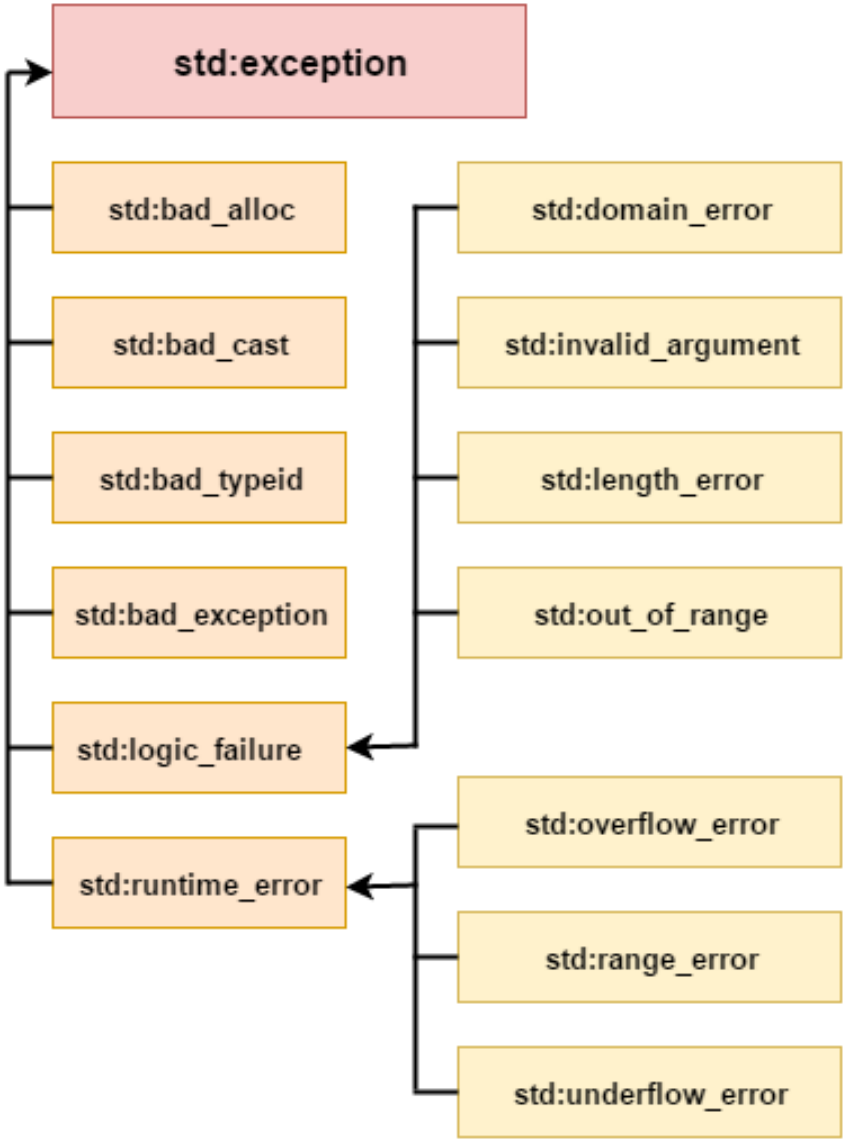
Ponovno prijavljivanje izuzetaka (*rethrowing*)

- Ponekad se u bloku za obradu izuzetaka može da se javi potreba da se izuzetak prosledi dalje, tada se navodi throw naredba bez argumenta:

- Primer:

```
catch(ExcType ex)
{
        // Obrada izuzetka
        throw;
}
```

Standardni izizeci u programskom jeziku C++



Osnovna klasa za predstavljanje izuzetaka u programskom jeziku C++

```
class exception
{
  public:
      exception () noexcept;
      exception (const exception&) noexcept;
      exception& operator= (const exception&) noexcept;
      virtual ~exception();
      virtual const char* what() const noexcept;
}
```

Korisnički definisani tipovi za predstavljanje izuzetaka

- Preporučuje se da se i korisnički izuzeci izvode iz klase exception
- Ovako kreirana hijerarhija izuzetaka omogućava da se pojedini tipovi izuzetaka mogu obrađivati :
  - pojedinačno
  - u srodnim grupama
  - svi zajedno (ali ipak ne kao ...)

Izuzeci u programskom jeziku Java

Obrada izuzetaka u programskom jeziku Java

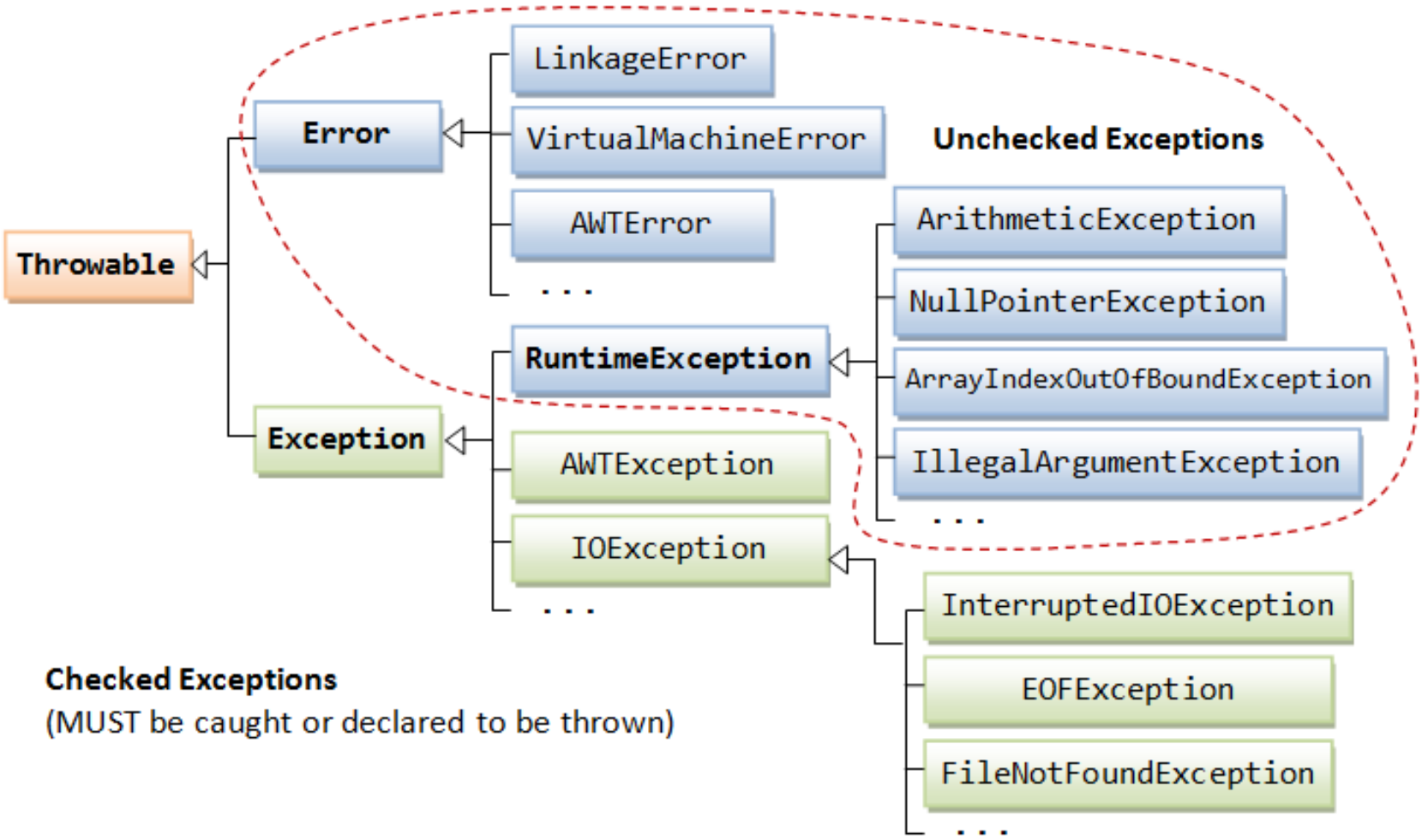
```
try {
       // Kod koji može da prijavi izuzetke
}
catch(ExceptionType1 ex) {
     // Obrada izuzetka tipa ExceptionType1
}
⋮
catch(ExceptionTypeN ex) {
     // Obrada izuzetka tipa ExceptionTypeN
}
finally {
     // Deo koji se izvršava svejedno da li je
     // izuzetak prijavljen ili ne
}
```

Namena *finally* bloka

- Da oslobodi resurse koji su eventualno zauzeti pre nego što je izuzetak prijavljen

```
File file = null;
try {
   file = new File("Datoteka")
   file.write("foo");
   file.write("bar");
}
catch (IOException e){
   // obrada izuzetka
   return;     //izvršava se finally blok
}
finally {
   if (file != null)
         file.close();
}
}
```

Standardni izuzeci u Javi



Standardni izuzeci u Javi

- Tip objekat koji se prijavljuje kao izuzetak mora da bude direktno ili indirektno izveden iz klase Throwble.
- Klasa Error – osnovna klasa za predstavljanje skupa izuzetaka koje uglavnom prijavljuje JVM i koje obavezno izazivaju prekid rada aplikacije. Zato prihvatanje i obrada ovih tipova izuzetaka u programu nije obavezna.
- Klasa Exception – osnovna klasa za predstavljanje izuzetaka koji se prijavljuju u programu – mogu da budu izazvani pogrešnim podacima, problemima u radu U/I uredjaja ili greškom programera.
- Klasa RuntimeException – za predstavljanje izuzetaka koji mogu da nastanu tokom izvršenja programa i uglavnom su posledica propusta programera. Izuzeci izvedeni iz ove klase, takodje, ne moraju biti obradjeni u programu.

Checked & Unchecked exceptions

- Unchecked – U fazi kompajliranja se ne proverava da li su izuzeci ovog tipa obrađeni
  - Izuzeci izvedeni iz klasa Error i RuntimeException
- Checked – U fazi kompajliranja se proverava da li su izuzeci ovog tipa obrađeni
  - Svi izuzeci izvedeni iz klase Exception, osim onih izvedenih iz klase RuntimeException.

Kreiranje sopstvenih klasa za predstavljanje izuzetaka

- Obavezno je da budu direktno ili indirektno izvedene iz klase Exception.
- Primer:

```
public class MyException extends Exception
{
  public MyException()
   {
     super( "MyException message");
   }
}
```

Propagiranje izuzetka u Javi

- Ukoliko funkicija prijavljuje izuzetak iz skupa Checked, u zaglavlju funkcije se obavezno navodi skup tipova izuzetaka koje ona prijavljuje (da bi kompajler mogao da proveri da li su obrađeni).
- Skup Checked tipova izuzetaka koje funkcija prijavljuje se navode nakon zaglavlja u klauzuli *throws:*

```
throws  NizTipova;
```

- Primer:

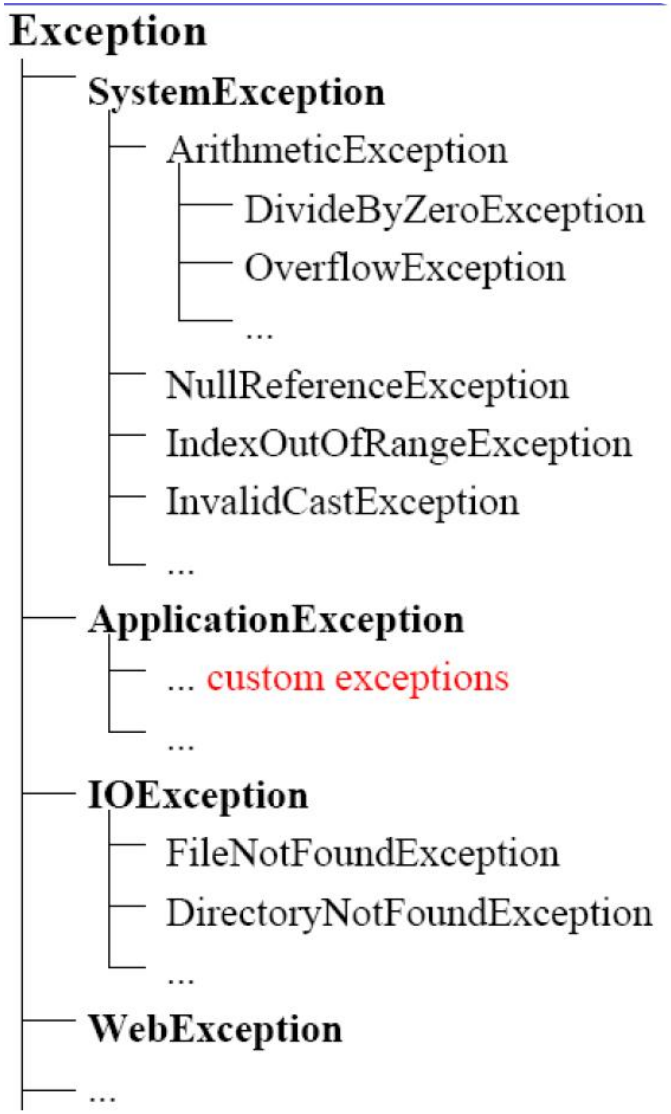
```
void ReadData() throws IOException
{


}
```

Sličnosti i razlike u odnosu na C++

- Sličnosti:
  - Format naredbe za prijavu izuzetaka isti
  - Struktura za obradu izuzetaka sadrži *try-catch* delove identične onima u C++-u
- Razlike:
  - Izuzetak ne može biti proizvoljnog tipa
  - Struktura za obradu izuzetaka može da sadrži dodatni *finally* deo
  - U zaglavlju funkcija koja prijavljuju izuzetke Checked tipa klauzula *throws* je **obavezna**

Izuzeci u programskom jeziku C#

Standardni izuzeci u programskom jeziku C#



Sopstvene klase za predstavljanje izuzetaka

- Nasleđuju klasu ApplicationException.

Sličnosti i razlike u odnosu na Javu

- Sličnosti:
  - Format naredbe za prijavu izuzetaka isti
  - Struktura za obradu izuzetaka ***try-catch-finally*** identična
  - Objekti koji se prijavljuju kao izuzeci pripadaju standardnim klasama za predstavljanje izuzetaka ili klasama izvedenim iz tih standardnih.
- Razlike:
  - Kod propagacije izuzetaka, u zaglavlju funkcija koja prijavljuju izuzetke se ne navodi lista tipova izuzetaka koje funkcja može da prijavi