Методика наставе рачунарства

Владимир Филиповић
vladaf@matf.bg.ac.rs

Изузетци у програмском језику Јава

Владимир Филиповић
vladaf@matf.bg.ac.rs

Изузетци

- Јава користи изузетке као начин сигнализирања озбиљних проблема приликом извршавања програма.
- Стандардне класе (класе из ЈДК-а) их интензивно користе.
- Пошто они настају када у Јава програмима ствари крену наопако, а пре или касније тако нешто се деси, изузетке треба узети у озбиљно разматрање када се дизајнира апликација и када се пишу програми.
- Разлог зашто изузецима до сада није посвећено више пажње је што је за њих потребно добро разумевање класа и механизма наслеђивања.

Изузетци (2)

- Изузетак обично сигнализира грешку или неки посебно необичан догађај у програму који заслужује посебну пажњу.
- Уколико се усред кода који рукује нормалним операцијама програма, истовремено рукује мноштвом често необичних грешака, структура програма ускоро постаје компликована и тешка за разумевање.
- Главна корист од изузетака јесте што они раздвајају код који обрађује грешке од кода који се извршава када ствари теку глатко.
- Други позитиван аспект изузетака јесте што обезбеђују механизам присиле да се реагује на одређене врсте грешака.

Изузетци (3)

- Важна идеја коју треба схватити је да не треба све грешке у програмима сигнализирати изузецима – само неуобичајене или катастрофалне.
- На пример, ако корисник не унесе исправан улазни податак, за то не треба користити изузетке, јер се ради о уобичајеном догађају.
- Разлог за такву одлуку је што руковање изузецима укључује много додатног процесирања, па ће програм који рукује изузецима већину времена бити знатно спорији него што би морао да буде.
- У Јави, изузетак је објекат који се креира када се деси ненормална ситуација у нашем програму.
- Објекат-изузетак садржи поља (атрибуте) у која се смештају информације о природи проблема.

Природно-математички факултет

vladaf@matf.bg.ac.rs

Изузетци (4)

- За изузетак се каже да је «подигнут» или «избачен» (енг. thrown).
- Објекат који идентификује изузетну ситуацију тј. изузетак се подиже као аргумент у специфичном делу кода које се пише баш да би руковало том врстом проблема.
- За код који прима објекат изузетак као параметар се каже да га «хвата» (енг. catch).

Изузетци (5)

Ситуације које узрокују изузетке су прилично разноврсне, али спадају у четири категорије:

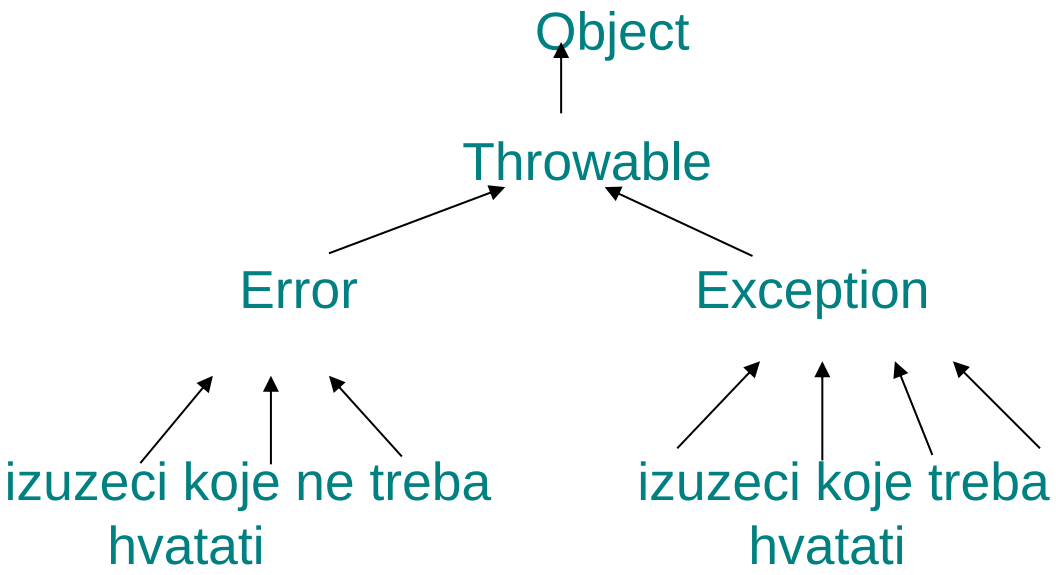
- грешке кода или података (неисправан покушај кастовања објекта, коришћење индекса који је изван граница за тај низ, дељење целог броја нулом);
- изузеци стандардних метода (нпр. избацивање StringIndexOutOfBoundsException изузетака);
- избацивање наших сопствених изузетака;
- Јава грешке (обично последица грешке у нашем програму).

Типови изузетака

Изузетак је увек објекат неке потклассе стандардне класе Throwable. То важи и за изузетке које сами дефинишемо, као и за стандардне изузетке.

Две директне потклассе класе Throwable – класа Error и класа Exception – покривају све стандардне изузетке.

Обе ове класе имају потклассе за специфичне изузетке.



Изузетци типа Error

Изузеци дефинисани класом Error и њеним поткласама карактеришу се чињеницом да се од нас не очекује да предузимамо ништа, не очекује се да их хватамо.

Класа Error има три директне потклассе:

- ThreadDeath – избацује се када се нит која се извршава намерно стопира.
- LinkageError – озбиљни проблеми са класама у програму, нпр. некомпатибилност међу класама или покушај креирања објекта непостојећег класног типа су неке од ствари које могу изазвати избацивање изузетка овог типа.
- VirtualMachineError – садржи четири потклассе изузетака који се избацују када се деси катастрофални пад JVM.

Изузетци типа Error (2)

Изузеци који одговарају објектима класа изведених из LinkageError и VirtualMachineError су резултат катастрофалних догађаја и услова.

У таквим ситуацијама обично све што ми можемо да урадимо јесте да прочитамо поруку о грешци која се генерише када се избаци изузетак, посебно у случају LinkageError изузетка и да покушамо да схватимо шта је у нашем коду могло да изазове проблем.

Изузетци RuntimeException

За скоро све изузетке представљене поткласама класе Exception, мора се у програм укључити код који ће руковати њима уколико наш код може изазвати њихово избацивање

Ако метод у нашем програму може да генерише изузетак типа који има Exception као надкласу, мора се или руковати изузетком унутар тог метода или регистровати да тај метод може избацити такав изузетак. У супротном, програм се неће превести.

Разлог због ког се RuntimeException изузетци другачије третирају и преводилац допушта да их игноришемо је тај што они генерално настају због озбиљних грешака у нашем коду.

У већини случајева можемо да урадимо мало да поправимо ситуацију.

Изузетци RuntimeException (2)

Међутим, у неким контекстима за неке од ових изузетака, то није увек случај и можда желимо да укључимо код за њихово препознавање.

Потклассе класе RuntimeException дефинисане у стандардном пакету java.lang су:

- ArithmeticException
- IndexOutOfBoundsException
- NegativeArraySizeException
- NullPointerException
- ArrayStoreException, ClassCastException, IllegalArgumentException, SecurityException, IllegalMonitorStateException, IllegalStateException, UnsupportedOperationException

Остале поткласе Exception

- За све остале класе изведене из класе Exception, преводилац ће проверити да ли је руковање изузетком извршено у методу где је изузетак могао бити избачен или је, алтернативно, назначено да метод може избацити такав изузетак. Уколико није урађено ни једно ни друго, тада се програм неће превести.
- Изузев неколико врста изузетака који имају RuntimeException као основу, сви изузеци избачени од метода Јавине библиотеке класа захтевају одговарајуће руковање.

Руковање изузетцима

Дакле, ако дати програмски код може избацити изузетке који нису типа Error или RuntimeException (и њихових поткласа – што се подразумева), онда се мора нешто предузмети поводом тога.

Кад год се пише програмски код који може да избаци изузетак, јавља се избор:

- може се креирати код унутар метода за руковање датим избаченим изузетком,
- може игнорисати изузетак у том методу и омогућити да метод који садржи код који може избацити изузетак проследи тај изузетак у позивајући метод

Руковање изузетцима (2)

- Претпоставимо да имамо метод који може избацити изузетак који није типа потклассе RuntimeException нити Error класе.
- Нека је изузетак нпр. типа IOException.
- Ако изузетак није ухваћен и није завршено са његовом обрадом унутар метода, најмање што морамо да урадимо јесте да декларишемо да може бити избачен изузетак. Како се то ради?
- Једноставно се дода throws клауза у дефиницију метода нпр.

```
double myMetod() throws IOException{...}
double myMetod() throws IOException,FileNotFoundException {...}
```

- Као што претходни примери показују, да декларишемо да наш метод може избацити изузетке, само додамо кључну реч throws након листе параметара метода. Затим се наводи листа класа за изузетке који могу бити избачени, раздвојених запетама.

Руковање изузетцима (3)

Ако неки други метод позива овај метод, он мора да узме у обзир изузетке које овај може избацити, па или их обрађивати или и он декларисати да избацује изузетке истог типа.

Уколико се не уради ни једно ни друго, преводилац ће то утврдити и доћи че до гршке превођења, па се Јава код неће превести у бајт-код.

Руковање изузетцима (4)

Ако желимо да рукујемо изузецима тамо где се они десе, потребно је укључити три врсте блокова кода у метод који рукује изузецима, и то су:

- try блок – обухвата код где се може јавити један или више изузетака. Код који може да избаци изузетак који желимо да ухватимо мора бити у try блоку;
- catch блок – обухвата код који је намењен да рукује изузецима одређеног типа који могу бити избачени у придруженом try блоку;
- finally блок – увек се извршава пре него се метод заврши, без обзира да ли је било који изузетак избачен у try блоку или није.

try блок

Када треба да се ухвати изузетак, код метода који може избацити изузетак мора бити обухваћен try-блоком.

Код који може изазвати изузетке не мора бити у try-блоку, али онда метод неће бити способан да ухвати ниједан од избачених изузетака, па у декларацији метода треба бити истакнуто да тај метод може избацити типове изузетака који нису ухваћени.

try-блок чини кључна реч try за којом следи пар витичастих заграда које окружују код који може избацити изузетак.

```
try
{
    // kod koji moze izbaciti jedan ili vise izuzetaka
}
```

try-блокови су неопходни и када је потребно да се ухвате изузетци типа Error ili RuntimeException (они се лако генеришу).

catch блок

- Код за руковање изузетком датог типа се ограђује catch-блоком.
- catch-блок се мора налазити непосредно иза try-блока који садржи код који може избацити тај одређени изузетак.
- catch-блок се састоји од кључне речи catch праћене једним параметром унутар облих заграда којим се идентификује тип изузетка којим блок рукује. Ово прати код за руковање изузетком који се налази унутар пара витичастих заграда:

```
try
{
        // kod koji moze izbaciti jedan ili vise izuzetaka
}
catch(ArithmeticException e)
{
        // kod za rukovanje izuzetkom tipa ArithmeticException
}
```

catch блок (2)

Овај catch блок рукује само изузецима типа ArithmeticException.

То повлачи да је то једина врста изузетака која може бити избачена у try-блоку. Ако могу бити избачени и други, ово се неће искомпајлирати.

Генерално, параметар за catch блок мора бити типа Throwable или неке њене потклассе. Ако класа коју задамо као параметар има потклассе, од catch блока се очекује да процесира изузетке тог типа, али и свих потклаcа тог типа.

Вишеструки catch блок

Ако try-може да избаци неколико различитих врста изузетака, тада је потребно поставити више catch блокова за руковање њима након try-блока.

Пример:

```
try
{
   // kod koji moze izbaciti izuzetke...
}
catch(ArithmeticException e)
{
   // kod za rukovanje ArithmeticException izuzecima
}
catch(IndexOutOfBoundsException e)
{
   // kod za rukovanje Index... izuzecima
}
// Izvrsavanje se nastavlja ovde ...
```

Вишеструки catch блок (2)

У претходном примеру изузеци типа ArithmeticException биће хватани првим catch блоком, а типа IndexOutOfBoundsException другим.

Наравно, ако се избаци изузетак типа ArithmeticException, биће извршен само код тог catch блока. Када се он заврши, извршавање се наставља наредбом након последњег catch блока.

Редослед catch блокова може бити од значаја. Када се избаци изузетак, он ће бити ухваћен првим catch блоком чији је параметар истог типа као изузетак или је типа неке његове надкласе.

Ако постоји хијерархија, редослед catch-блокова би требало да буде: најизведенији тип прво, најосновнији тип на крају. Иначе, код се неће превести.

Природно-математички факултет

vladaf@matf.bg.ac.rs

finally блок

Природа изузетака је таква да се извршавање try блока прекида по избацивању изузетка без обзира на значај кода који следи тачку у којој је изузетак избачен.

То уводи могућност да изузетак остави ствари у незадовољавајућем стању. На пример, можемо имати отворен фајл и пошто је избачен изузетак, не извршава се код за затварање фајла.

finally-блок обезбеђује средство да се ”почисти” на крају извршавања try-bloka. Finally блок се користи када треба да се буде сигурно да се неки одређени код извршава пре краја метода, без обзира који изузеци су избачени унутар придруженог try-блока.

finally--блок се извршава увек, без обзира да ли су или не избачени изузеци за време извршавања придруженог try блока.

finally блок (2)

Структура finally блока је:

```
finally
{
    // odgovarajuci kod ...
}
```

Управо као и catch блок, тако је и finally блок придружен одређеном try блоку и мора бити смештен непосредно након catch блокова за тај try блок.

Ако нема catch блокова, finally блок се смешта непосредно након try блока. Иначе се програм неће успешно превести.

Уколико је коришћењем return наредбе враћена нека вредност унутар finally блока, то поништава return наредбу која је евентуално извршена у try блоку.

finally блок (3)

```
try{
        // Kod koji moze izbaciti izuzetke ...
}catch(ExceptionType1 e){
        // ...
}catch(ExceptionType2 e){
        // ...
        // ako je potrebno, jos catch blokova ...
}finally{
        // kod koji se uvek izvrsava nakon try-bloka
}
```

Није могуће да постоји само try-блок, већ њега увек мора да прати бар један од catch и finally блокова.

Изузетци који нису ухваћени могу бити избачени било где у телу метода, у делу кода који није ограђен try-блоком.

Природно-математички факултет

vladaf@matf.bg.ac.rs

Пропагирање изузетака

У многим ситуацијама, када се у неком методу појави изузетак, програмер може да одлучи да тај изузетак не обрађује на у том методу већ да га омогући да изузетак пропагира у позивајући метод.

Мотивација за такву одлуку је што је позивајући метод у принципу свеснији контекста у ком је изузетак настао, па се на том нивоу лакше може одлучити које акције треба предузети.

Пропагирање изузетка у метод-позивалац се реализује помоћу кључне речи throws иза које следи листа изузетака којима је допуштено пропагирање.

Избацивање изузетака

У многим ситуацијама, када неки метод ухвати изузетак, имплементирањем одговарајуће catch клаузе, позивајући програм можда мора да зна да се то десило.

Ако изузетак који смо ухватили треба да проследимо позивајућем програму, можемо да га поново избацимо из унутрашњости блока catch, користећи throw наредбу, на пример:

```
try{
        // ...
}catch(ArithmeticException e){
        // obrada ovog izuzetka
        throw e;
}
```

Код throw наредбе је дата кључна реч throw за којом следи објекат типа изузетка који се избацује.

Избацивање изузетака (2)

Програмер може одлучити да избаци изузетак кад год нађе за сходно, чак и у „нормалној“ ситуацији.

Међутим, треба нагласити да су изузетци неефикасни и да их треба искључиво користити за „нерегуларне“ ситуације, а не за реализацију делова „нормалне“ пословне логике.

Природно-математички факултет

vladaf@matf.bg.ac.rs

Захвалница

Велики део материјала који је укључен у ову презентацију је преузет из презентације коју је раније (у време када је он држао курс Објектно орјентисано програмирање) направио проф. др Душан Тошић.

Хвала проф. Тошићу што се сагласио са укључивањем тог материјала у садашњу презентацији, као и на помоћи коју ми је пружио током конципцирања и реализације курса.

Надаље, један део материјала је преузет од колегинице Марије Милановић.

Хвала Марији Милановић на помоћи у реализацији ове презентације.