Објектно орјентисано програмирање

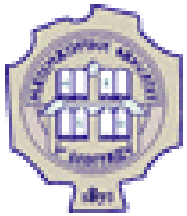
Владимир Филиповић
vladaf@matf.bg.ac.rs
Александар Картељ
kartelj@matf.bg.ac.rs

Изузеци и тврдње у програмском језику Јава

Владимир Филиповић
vladaf@matf.bg.ac.rs
Александар Картељ
kartelj@matf.bg.ac.rs

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



Изузеци

- 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, 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.

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Изузеци (2)

- Изузетак обично сигнализира грешку или неки посебно необичан догађај у програму који заслужује посебну пажњу.
- 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.

Математички факултет

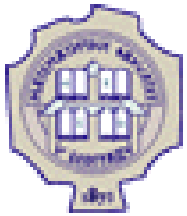
{vladaf,kartelj}@matf.bg.ac.rs

Изузеци (3)

- 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!
- Разлог је што руковање изузецима укључује много додатног процесирања што успорава целокупан програм.
- Изузетак је **објекат** са информацијама о проблему који се креира када се деси ненормална ситуација у нашем програму.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



Изузеци (4)

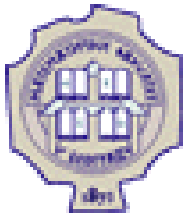
- За изузетак се каже да је «подигнут» или «избачен» (енг. thrown).
- За код који прима објекат изузетак као параметар се каже да га «хвата» (енг. catch).
- Једноставан пример кода који избацује изузетак:

```
public class JednostavanPrimer {

	public static void main(String[] args) {
		try {
			int a[] = new int[2];
			System.out.println("Pristupam elementu :" + a[3]);
		} catch (ArrayIndexOutOfBoundsException e) {
			System.out.println("Izuzetak izbacen :" + e);
		}
	}
}
```

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



Изузеци (5)

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

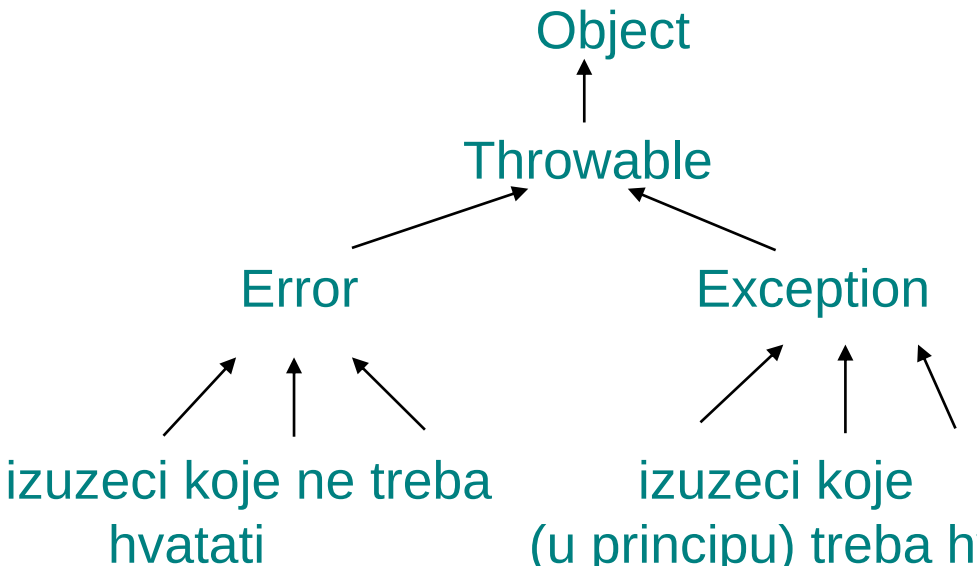
1. грешке кода или података:
   - неисправан покушај кастовања објекта,
   - коришћење индекса који је изван граница за тај низ,
   - дељење целог броја нулом;
2. изузеци стандардних метода (нпр. избацивање StringIndexOutOfBoundsException изузетака);
3. избацивање кориснички дефинисаних изузетака;
4.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

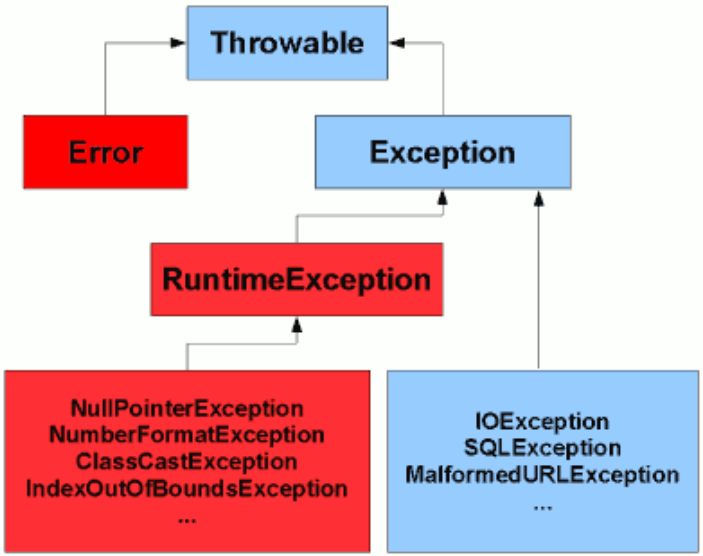
Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Типови изузетака

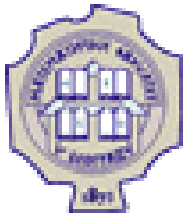
- Изузетак је увек објекат неке поткласе стандардне класе Throwable.
  - То важи и за изузетке које сами дефинишемо, као и за стандардне изузетке.
- Две директне поткласе класе Throwable – класа Error и класа Exception – покривају све стандардне изузетке.
- Обе ове класе имају поткласе за специфичне изузетке.





Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

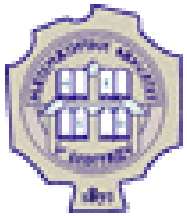


Изузеци типа Error

- Изузеци дефинисани класом Error и њеним поткласама карактеришу се чињеницом да се од програмера не очекује да предузима ништа, не очекује се да их хвата.
- 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 и слично.
  - VirtualMachineError – садржи четири потклассе изузетака који се избацују када се деси катастрофални пад JVM.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

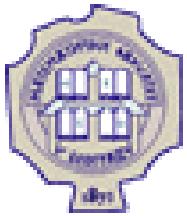


Изузеци типа Error (2)

- Изузеци који одговарају објектима класа изведених из LinkageError и VirtualMachineError су резултат катастрофалних догађаја и услова.
  - У таквим ситуацијама обично све што програмер може да уради јесте да прочита поруку о грешци која се генерише када се избаци изузетак, посебно у случају LinkageError изузетка.
  - На основу поруке треба да покушава да се схвати шта је у написаном коду могло да изазове такав проблем.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



Изузеци 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
- Изузетак су објекти класе RuntimeException.
  - Преводилац допушта да их програмер игнорише јер они генерално настају због озбиљних грешака у написаном програмском коду.
  - Изузеци класе RuntimeException, дакле, указују на то да је нешто лоше у самој логици написаног програма.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Изузеци RuntimeException (2)

- У неким контекстима за неке од ових изузетака је можда потребно да програмер укључи код за њихово препознавање.
  - То је већ показано на примеру IndexOutOfBoundsException изузетка, где покушавамо да приступимо елементу ван граница низа.
  - Ово је, наравно, неприродан пример који јасно указује на грешку у логици програмирања.
- Потклассе класе RuntimeException су:
  - ArithmeticException
  - IndexOutOfBoundsException
  - NegativeArraySizeException
  - NullPointerException
  - ArrayStoreException, ClassCastException, IllegalArgumentException, SecurityException, IllegalMonitorStateException, IllegalStateException, UnsupportedOperationException

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Остале поткласе Exception

- За све остале класе изведене из класе Exception, преводилац ће проверити да ли је испуњена једна од ове две ствари у оквиру методе која може да избаци изузетак:
  1. Хватање изузетка (try-catch блок);
  2. Прослеђивање изузетка (наредба throws) надметоди, односно методи која је позвала нашу методу.
- Уколико није урађено ни једно ни друго, тада се програм неће превести.
- Дакле, сви изузеци који нису типа Error или RuntimeException се морају разрешити.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Руковање изузецима (2)

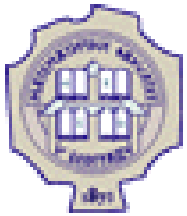
- Претпоставимо да наш метод позива неки метод који може избацити изузетак који није типа потклассе RuntimeException нити Error класе.
- Нека је изузетак нпр. типа IOException.
- Најмање што морамо да урадимо јесте да декларишемо да може бити избачен изузетак. Како се то ради?
- Једноставно се дода throws клауза у дефиницију метода нпр.

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

- Дакле, само се дода кључна реч throws и листа изузетака који могу бити избачени, раздвојених запетама.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



Руковање изузецима (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 ће до грешке превођења, па се Јава код неће превести у бајт-код.

Математички факултет

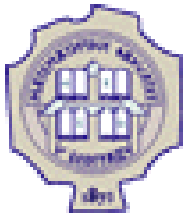
{vladaf,kartelj}@matf.bg.ac.rs

Руковање изузетцима (4)

- Ако се одлучи да се рукује изузецима тамо где се они десе, потребно је укључити три врсте блокова кода у метод који рукује изузецима, и то су:
  1. 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;
  2. catch блок – обухвата код који је намењен да рукује изузецима одређеног типа који могу бити избачени у придруженом try блоку;
  3. finally блок – увек се извршава пре него се метод заврши, без обзира да ли је било који изузетак избачен у try блоку или није.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



try блок

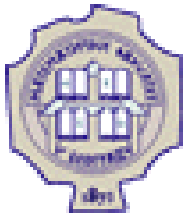
- Када треба да се ухвати изузетак, код метода који може избацити изузетак мора бити обухваћен try-блоком.
- Код који може изазвати изузетке не мора бити у try-блоку, па у декларацији метода треба бити истакнуто да тај метод може избацити типове изузетака који нису ухваћени.
- try-блок чини кључна реч 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).

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



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 tipaArithmeticException
}
```

Математички факултет

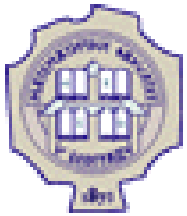
{vladaf,kartelj}@matf.bg.ac.rs

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, претходни код се неће успешно превести.
- Генерално, параметар за catch блок мора бити типа Throwable или неке њене потклассе.
- Ако класа која се зада као параметар catch блока има потклассе, од catch блока се очекује да процесира изузетке тог типа, али и свих потклаcа тог типа.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



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 ...
```

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

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 или је типа његове надкласе.
- Ако постоји хијерархија, редослед catch-блокова би требало да буде: најизведенији тип прво, најосновнији тип на крају.

Математички факултет

{vladaf,kartelj}@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, може се догодити да се отвори датотека и да се, пошто је избачен изузетак, не извршава се код за затварање те датотеке.
- finally-блок обезбеђује средство да се ”почисти” на крају извршавања try-bloka
- finally--блок се извршава увек, без обзира да ли су или не избачени изузеци за време извршавања придруженог try блока.

Математички факултет

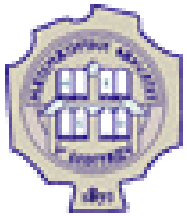
{vladaf,kartelj}@matf.bg.ac.rs

finally блок (2)

- 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.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



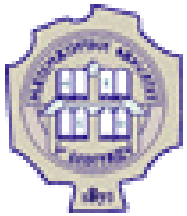
finally блок (3)

Структура комплетне try-catch-finally наредбе:

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

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

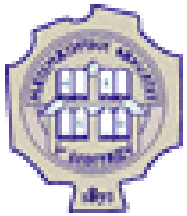


Пропагирање изузетака

- 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 га 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.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs



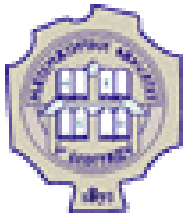
Избацивање изузетака

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

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs



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.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Тврдње

- Тврдње допуштају да се тестира исправност било које од претпоставки која је направљена приликом писања програма
- Тврдње се у Јави постављају помоћу наредбе assert
- Током извршавања тдње, претпоставка је да је она тачна. Ако тврдња није тачна, тада JBM избацује грешку типа AssertionError
- Тврдње се најчешће користе за потребе тестирања приликом развоја програма и њихово процесирање се по правилу искључује када програм пређе продукцију
- Јединични тестови су у великој мери потписнули тврдње
- Наредба assert се може јавити у два облика:

```
assert logicki-izraz;

assert logicki-izraz : niska-izraz;
```

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Тврдње (2)

- Омогућавање тврдњи
  - Подразумевано је постављено да су тврдње онемогућене
  - Покретање Јава програма тако да буду омогућене тврдње захтева укључивање –ea опције за JBM.
  - На пример, покретање main метода у класи Test садржаној у датотеци Test.java са омогућеним тврдњама постиже се са:

    java –ea Test
- Онемогућавање тврдњи
  - Покретање Јава програма тако да буду онемогућене тврдње захтева укључивање –da опције за JBM.
  - На пример, покретање main метода у класи Test садржаној у датотеци Test.java са онемогућеним тврдњама постиже се са:

    java –da Test

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Тврдње (3)

- Основни разлог за коришћење тврдњи је жеља програмера да провери да ли важе претпоставке које је увео и на којима се ослања његов програмски код
- Тврдњама се обезбеђује:
  - да програмски код који изгледа недоступан заиста и јесте недоступан
  - да претпоставке описане у коментарима заиста важе
  - да се не догађа да у switch наредби извршавање долази у default грану
  - да се потврди исправност стања објекта
- Тврдње се обично постављају или на почетку извршења метода (предуслови) или по завршетку позива метода (постуслови)

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Тврдње (4)

- У принципу, тврдње се користе за проверу логички немогућих ситуација, тј. ситуација које не би смеле да се догоде и из којих нема регуларног опоравка
- Стога, за разлику од рада са „нормалним“ изузетцима, нема смисла да се они обрађују током извршавања када проблем буде детектован
- Обично се тврдње искључују у продукционом коду
- Где се обично користе тврдње:
  - За проверу аргумената прослеђених приватним методама
  - За проверу исправностипроласка кроз наредбе гранања
  - За проверу важења датих услова на почетку метода

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Тврдње (5)

- Где у принципу не треба користити тврдње:
  - Тврдње не треба да замене поруке о грешкама
  - Тврдње не треба да се користе за контролу аргумената у јавним методама, јер тада аргументе може проследити било ко. Тада проверу аргумената треба реализовати кроз механизам изузетака
  - Тврде не треба користити за проверу аргумената командне линије, јер не можемо знати унапред ко ће и са којим аргументима покретати дати програм

Математички факултет

{vladaf,kartelj}@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 презентацију, 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.