TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

WELCOME TO TECHPROED JAVA TUTORIAL

Testi baslatmak icin asagidaki adımları takip ediniz

Go to www.socrative.com

Click on Login

Click on Student Login

***Room Name*: A L P T E K I N 3 5 2 3**

Kayıtta kullandığınız ismi tam olarak yazınız

Time: 11 Minutes

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Exception Nedir?

Hersey güzel, problem yok.
No exception

Sürücü exception'i halletmeye çalışıyor yani; handling exception

Turist exception'ı halledemiyor ve yardım istiyor yani; throwing exception

Java code çalıştırılırken, farklı error'lar oluşabilir:

1) **Code'u yazan hatalı yazmış olabilir**
2) **Kullanıcı hatalı data girişi yapıyor olabilir**
3) **Öngörülemeyen hatalar oluşabilir**

Hata oluştuğunda Java çalısmayı durdurur ve console'a bir hata mesajı yazdırır.
Buna exception atma manasına gelen "throw an exception" denir.

Exception Types

1) Checked Exceptions *(Compile time exceptions)*: **Checked Exception'lar compile-time'da check edilirler.**

```
class Example {

    public static void main(String args[]){

        FileInputStream fis = new FileInputStream( name: "B:/myfile.txt");
        int k;

        while(( k = fis.read() ) != -1){

            System.out.print((char)k);
        }

        fis.close();

    }
}
```

Note: Checked Exception'lar kesinlikle code yazılırken halledilmelidirler.

Meşhur Compile Time Exception'lar

1) FileNotFoundException: Code icinde kullanmak istediğimiz dosyaya herhangi bir sebeple ulaşamadığımız zaman atılan exception'dır.

2) IOException: (IO stands for InputOutput)Butun Input veya Output durumlarında oluşabilecek tüm hatalarda atılan exception'dır.

Note: FileNotFoundException, IOException'ın child'ıdır.

“Checked Exceptions” lar nasıl handle edilir ?

1)“throws” keyword kullanılabilir.

```
class Example {

    public static void main(String args[]) throws IOException {

        FileInputStream fis = new FileInputStream( name: "B:/myfile.txt");
        int k;

        while(( k = fis.read() ) != -1)
        {
            System.out.print((char)k);
        }
        fis.close();
    }

}
```

2)“try-catch blocks” kullanılabilir

```
class Example {
    public static void main(String args[]) {

        FileInputStream file = null;

        try{

            file = new FileInputStream( name: "B:/myfile.txt");

        } catch(IOException e) {

            System.out.println("The specified file is not present at the given path");

        }

    }
}
```

“try-catch blocks” daha iyidir çünkü, daha anlaşılır mesaj verir

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

Try Catch Blocks

1)“try block” exception üretme ihtimali olan code içerir.
2) “try block”dan sonra her zaman “catch block” yazılır, “catch block” exception’i handle eder.
3) “try block”dan sonra 1’den fazla “catch block” kullanilabilir, istenirse “finally block” da kullanilabilir. “finally block” exception oluşsa da oluşmasa da çalışır.

```
class Example {
    public static void main(String args[]) {

        FileInputStream file = null;

        try{

            file = new FileInputStream( name: "B:/myfile.txt");

        }

    }
}
```

```
class Example {
    public static void main(String args[]) {

        FileInputStream file = null;

        try{

            file = new FileInputStream( name: "B:/myfile.txt");

        } catch(IOException e) {

            System.out.println("The specified file is not present at the given path");

        }

    }
}
```

```
class Example {
    public static void main(String args[]) {

        FileInputStream file = null;

        try{

            file = new FileInputStream( name: "B:/myfile.txt");

        } catch(IOException e) {

            System.out.println("The specified file is not present at the given path");

        } finally {

            System.out.println("Finally block runs everytime");

        }

    }
}
```

Exception yoksa catch block çalışmaz

finally block exception oluşsa da oluşmasa da çalışır mesela AWS’e connection’i kesme durumlarında kullanılır.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

4) Program yazarken bazı code'ların hata üretebileceğini öngörüyorsanız bu code'u try block'un icine koyup hatayı catch block kullanarak yakalayabilirsiniz.

```
class Example {
    public static void main(String args[]) {

        FileInputStream file = null;

        try{

            file = new FileInputStream( name: "B:/myfile.txt");

        } catch(IOException e) {

            System.out.println("The specified file is not present at the given path");

        }

    }
}
```

Bir dosyaya ulaşmak istediğinizde belki path yanlıştır veya belki dosya başka biri tarafından silinmiştir. Bundan dolayı try-catch block kullanmak iyidir.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION

5) Bir try block 1'den fazla catch block içerebilir.
Her catch block'da farklı bir exception yakalayabilirsiniz.

```java
class Example {
    public static void main(String args[]) {
        int num1, num2;
        try {
            /* We suspect that this block of statement can throw
             * exception so we handled it by placing these statements
             * inside try and handled the exception in catch block
             */
            num1 = 0;
            num2 = 62 / num1;
            System.out.println(num2);
            System.out.println("Hey I'm at the end of try block");
        }
        catch (ArithmeticException e) {
            /* This block will only execute if any Arithmetic exception
             * occurs in try block
             */
            System.out.println("You should not divide a number by zero");
        }
        catch (Exception e) {
            /* This is a generic Exception handler which means it can handle
             * all the exceptions. This will execute if the exception is not
             * handled by previous catch blocks.
             */
            System.out.println("Exception occurred");
        }
        System.out.println("I'm out of try-catch block in Java.");
    }
}
```

1) Eğer exception ilk catch block'da yakalanırsa sonraki catch block'lar çalışmaz.
2) Sonraki catch block'lar kesinlikle öncekilerin parent'i olmalıdır. Aksi takdirde Compile Time Error alırsınız.

What is output ?

```
String s = "";
try {
   s += "t";
} catch(Exception e) {
   s += "c";
} finally {
   s += "f";
}
s += "a";
System.out.print(s);
```

Sorular…

Doğru / Yanlış Soruları

1) try block catch block kullanmak zorundadir. (Yanlış)
2) finally block her zaman calisir. (Doğru)
3) try block ‘dan sonra 1 den fazla catch block kullanilabilir. (Doğru)
4) 1 den fazla catch block kullanirsaniz child exception once kullanilmalidir.(Doğru)

Açıklama Soruları

1) “FileNotFoundException” aciklayiniz.

2) “IOException” aciklayiniz.

Output Nedir?

```
public static void add(){
    String str = "";
    System.out.println(str.length());
}
```

Output Nedir?

```
public static void add(){
    String str = null;
    System.out.println(str.length());
}
```

Output Nedir?

```
class Example {

    public static void main(String args[]) {
        System.out.println(exceptions());
    }

    public static String exceptions() {
        String result = "";
        String v = null;
        try {
            try {
                result += "before";
                v.length();
                result += "after";
            } catch (NullPointerException e) {
                result += "catch";
                throw new RuntimeException();
            } finally {
                result += "finally";
                throw new Exception();
            }
        } catch (Exception e) {
            result += "done";
        }
        return result;
    }
}
```

Exception'larin console'a Yazdirilmasi

1)

```
class Example {

    public static void main(String args[]) {

        try {
             hop();
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    private static void hop() {
         throw new RuntimeException("cannot hop");
    }
}
```

2)

```
class Example {

    public static void main(String args[]) {

        try {
             hop();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    private static void hop() {
         throw new RuntimeException("cannot hop");
    }
}
```

3)

```
class Example {

    public static void main(String args[]) {

        try {
             hop();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void hop() {
         throw new RuntimeException("cannot hop");
    }
}
```

2) Unchecked Exceptions *(Run time exceptions)*: **Unchecked exception'lar compile time's check edilmezler. Exception'i kodu calistirdiktan sonra alirsiniz.**

```
class Example {
    public static void main(String args[])
    {
        int num1=10;
        int num2=0;

        int res=num1/num2;
        System.out.println(res);
    }
}
```

Note: Unchecked Exception'lar handle edilmek zorunda degildir.

Common Run Time (Unchecked Exception) Exceptions:

1) **ArithmeticException:** Sıfıra bölme yapma durumlarında oluşur.

```
System.out.println(12 / 0);
```

2) **ArrayIndexOutOfBoundsException:** Bir array'de olmayan index'e ulaşmak istediğimizde oluşur.

```
int arr[] = {a, b, c}   System.out.println( arr[12] );
```

3) **ClassCastException:** Uygun olmayan type casting durumlarında oluşur.

```
Object obj = new Object();
String sObj = (String)obj;
```

4) IllegalArgumentException:

```
public static void yumurtaSayisi(int tane) {
    if (tane < 0){
        throw new IllegalArgumentException( “Yumurta sayisi negatif olamaz”);
    }
    this.tane = tane;
}
```

5) NullPointerException: null bir object’in uzunluğuna ulaşmaya calistigimizda oluşur.

```
String name = null;
public void printLength(){
    System.out.println(name.length());
}
```

6) NumberFormatException: Rakamlarla oluşturulmayan bir String’i integer’a çevirmek isterken oluşur.

```
int num = Integer.parseInt("abc");
```

throw ile throws Farkları

```
void Demo() throws ArithmeticException, NullPointerException{

    throw new ArithmeticException();

}
```

1) throw method body'si icine yazılır, throws method parantezi ile curly brace'in arasına yazılır

2) throw'u method body'si icinde exception almak istediğimiz yerde kullanabiliriz throws ise mecburen method isminden sonra bir kere kullanilabilir

3) 1 throw sadece 1 tane exception üretir, 1 throws ile birden çok exception üretebilirsiniz yani ") throws NullPointerException, ArithmeticException { " mümkün

4) throw icin syntax ==> throw new NullPointerException;
   throws icin syntax ==> throws NullPointerException

finally keyword try-catch block ile beraber kullanilir ve her zaman calisir.
Genellikle bir kaynakla baglantinin kesilmesi gerektiği durumlarda kullanilir.

finalize() method Garbage Collector tarafından imha edilmesi gereken datalar imha edilmeden once calistirilir.

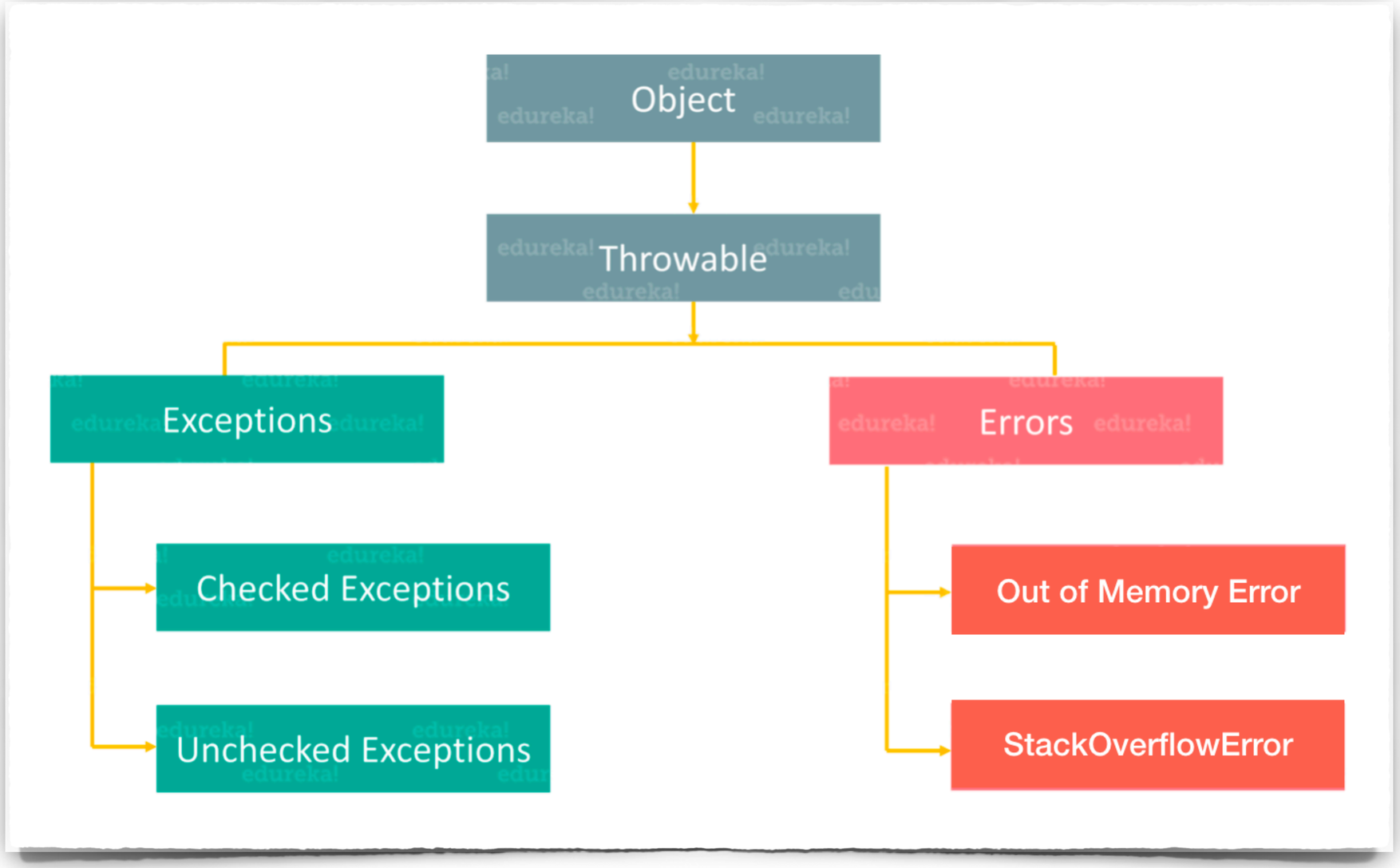
Garbage

Finalized Garbage

Object Destroyed

Final Variable	→	Degeri degistirilmeyecek variable'lar oluşturmak icin kullanılır.
Final Methods	→	Method Overridingi engellemek icin kullanılır.
Final Classes	→	Inheritance'i engellemek icin kullanılır.

TECHPROED
PROFESSIONAL TECHNOLOGY EDUCATION



Errors

1) Error'lar handle edilemezler.

2) Programın anormal bir sekilde sonlanmasına sebep olurlar.

3) Error'lar unchecked'dir ve genellikle run time'da oluşurlar.

4) Error'lara ornek; Out of Memory Error veya System Crash Error.

```
class Example {

    public static void main(String args[]) {

        for(int i=0; i>3; i--){
            System.out.println(i);
        }
    }
}
```

Sorular…

1) Which of the following pairs fill in the blanks to make this code compile? (Choose all that apply)

```
7: public void ohNo() _____ Exception {
8:   _____________ Exception();
9: }
```

A. On line 7, fill in `throw`

B. On line 7, fill in `throws`

C. On line 8, fill in `throw`

D. On line 8, fill in `throw new`

E. On line 8, fill in `throws`

F. On line 8, fill in `throws new`

2) Which exception will the following throw?

```
Object obj = new Integer(3);
String str = (String) obj;
System.out.println(str);
```

A. `ArrayIndexOutOfBoundsException`

B. `ClassCastException`

C. `IllegalArgumentException`

D. `NumberFormatException`

E. None of the above.

3) What will happen if you add the statement `System.out.println(5 / 0);` to a working `main()` method?

A. It will not compile.

B. It will not run.

C. It will run and throw an `ArithmeticException`.

D. It will run and throw an `IllegalArgumentException`.

E. None of the above.

4) Which of the following can be inserted in the blank to make the code compile? (Choose all that apply)

```
public static void main(String[] args) {
  try {
    System.out.println("work real hard");
  } catch (_____________________ e) {
  } catch (RuntimeException e) {
  }
}
```

A. `Exception`

B. `IOException`

C. `IllegalArgumentException`

D. `RuntimeException`

5)

What is the output of the following snippet, assuming a and b are both 0?

```
3:      try {
4:        return a / b;
5:      } catch (RuntimeException e) {
6:        return -1;
7:      } catch (ArithmeticException e) {
8:        return 0;
9:      } finally {
10:       System.out.print("done");
11:     }
```

A. `-1`

B. `0`

C. `done-1`

D. `done0`

E. The code does not compile.

F. An uncaught exception is thrown.

6) What is the output of the following program?

```
1:  public class Dog {
2:    public String name;
3:    public void parseName() {
4:      System.out.print("1");
5:      try {
6:        System.out.print("2");
7:        int x = Integer.parseInt(name);
8:        System.out.print("3");
9:      } catch (NumberFormatException e) {
10:       System.out.print("4");
11:     }
12:   }
13:   public static void main(String[] args) {
14:     Dog leroy = new Dog();
15:     leroy.name = "Leroy";
16:     leroy.parseName();
17:     System.out.print("5");
18:   } }
```

A. 12

B. 1234

C. 1235

D. 124

E. 1245

F. The code does not compile.

G. An uncaught exception is thrown.

Answers

1) B, D. In a method declaration, the keyword `throws` is used. To actually throw an exception, the keyword `throw` is used and a new exception is created.

2) B. The second line tries to cast an `Integer` to a `String`. Since `String` does not extend `Integer`, this is not allowed and a `ClassCastException` is thrown.

3) C. The compiler tests the operation for a valid type but not a valid result, so the code will still compile and run. At runtime, evaluation of the parameter takes place before passing it to the `print()` method, so an `ArithmeticException` object is raised.

4) C, E. Option C is allowed because it is a more specific type than `RuntimeException`. Option E is allowed because it isn't in the same inheritance tree as `RuntimeException`. It's not a good idea to catch either of these. Option B is not allowed because the method called inside the `try` block doesn't declare an `IOException` to be thrown. The compiler realizes that `IOException` would be an unreachable `catch` block. Option D is not allowed because the same exception can't be specified in two different `catch` blocks. Finally, option A is not allowed because it's more general than `RuntimeException` and would make that block unreachable.

5) E. The order of `catch` blocks is important because they're checked in the order they appear after the `try` block. Because `ArithmeticException` is a child class of `RuntimeException`, the `catch` block on line 7 is unreachable. (If an `ArithmeticException` is thrown in `try` try block, it will be caught on line 5.) Line 7 generates a compiler error because it is unreachable code.

6) E. The `parseName` method is invoked within `main()` on a new `Dog` object. Line 4 prints `1`. The `try` block executes and `2` is printed. Line 7 throws a `NumberFormatException`, so line 8 doesn't execute. The exception is caught on line 9, and line 10 prints `4`. Because the exception is handled, execution resumes normally. `parseName` runs to completion, and line 17 executes, printing `5`. That's the end of the program, so the output is `1245`.