PDP
IT-ACADEMY

Exception Handling

Reja:

1. Java Exceptions
2. try, catch, finally bloks
3. throw and throws keywords
4. try-with-resources
5. Create custom Exception

Exceptions

- Exception bu oldindan nazarda tutilmagan istisno holat bo’lib dastur bajarilish jarayonida yuzaga keladi va dasturning normal bajarilishiga to’sqinlik qiladi.
- Javada barcha error и exception lar **Throwable** klassidan voris olgan. Method ichida istisno holati sodir bo’lganda **`Throwable`** quyi klasslari toifasidagi object yaratiladi va runtime system ga berib yuboriladi. Ushbu object exception objecti deb yuritiladi.
- Exception object uning toifasi, dasturning o’sha paytdagi holati haqidagi ma’lumotlarga ega bo’ladi.
- Exception objectni yaratish va uni runtime system ga yuborish **throwing an exception** deb aytiladi.

Exception handling

Agar dastur bajarilish jarayonida exception objecti yaratiladigan bo’lsa bizda 2 ta yo’l mavjud:

- unga method doirasida ishlov berish;
- uni chaqirayotgan tomonga ishlov berish uchun berib yuborish.

Methodda aniq ko’rsatilishi kerak qanday istisno holatlariga ishlov beradi, qaysilariga esa yo’q.

Exception ga method doirasida ishlov berish uchun uni **try-catch** bloki orqali ushlab olib **catch** blokida amalga oshiriladi.

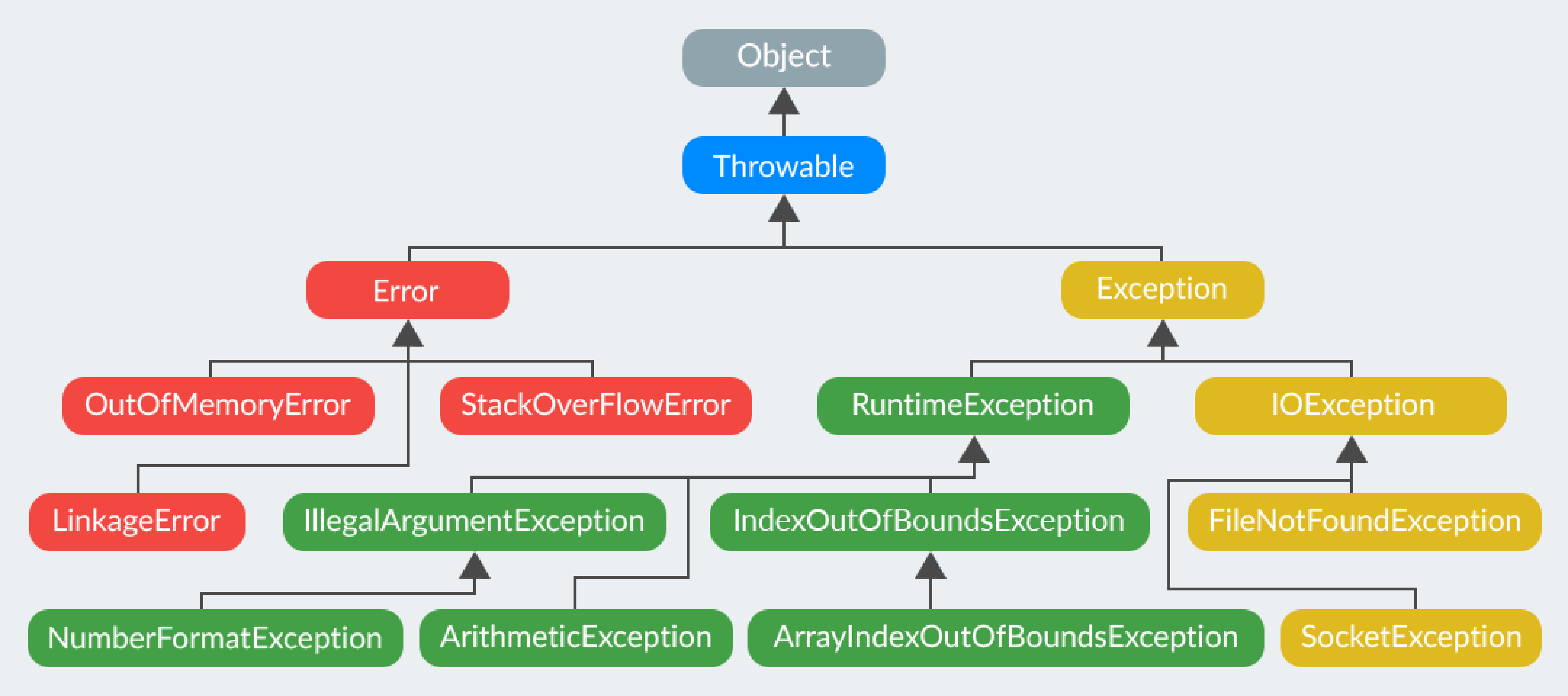
Exception ni chaqirayotgan tomonga berib yuborish methodni header qismida **throws** kalit so’zi orqali amalga oshiriladi.

Agarda Exception ga dastur doirasida ishlov berilmasa u JVM ga beriladi va odatda JVM dastur ishini to’xtatadi.

Exceptions

- Umuman Java da exception larning 3 xil turi mavjud:

  - **Checked (Compile time) exceptions**;
  - **Uncheked (Runtime) exceptions;**
  - **Errors.**


Object
Throwable
Error
Exception
OutOfMemoryError
StackOverFlowError
RuntimeException
IOException
LinkageError
IllegalArgumentException
IndexOutOfBoundsException
FileNotFoundException
NumberFormatException
ArithmeticException
ArrayIndexOutOfBoundsException
SocketException

Unhecked Exceptions

```
public static void main(String[] args) {
    int[] nums ={1,2,3};
      int a=nums[5];
}
```

Checked Exceptions

```
DateFormat dateFormat=new SimpleDateFormat("dd.MM.yyyy");

try {
    Date date=dateFormat.parse("02.08.2014");
} catch (ParseException e) {
    e.printStackTrace();
}
```

Java try catch finally blocks

- try block
- catch block
- finally block
- only try block is mandatory

try block

```
try {
    //application code
}
```

catch block

```
try {
    //code
}
catch(Exception e) {
    //handle exception
}
```

```
try {
    //code
}
catch(NullPointerException e) {
    //handle exception
}
catch(NumberFormatException e) {
    //handle exception
}
catch(Exception e) {
    //handle exception
}
```

PDP
IT-ACADEMY

finally block

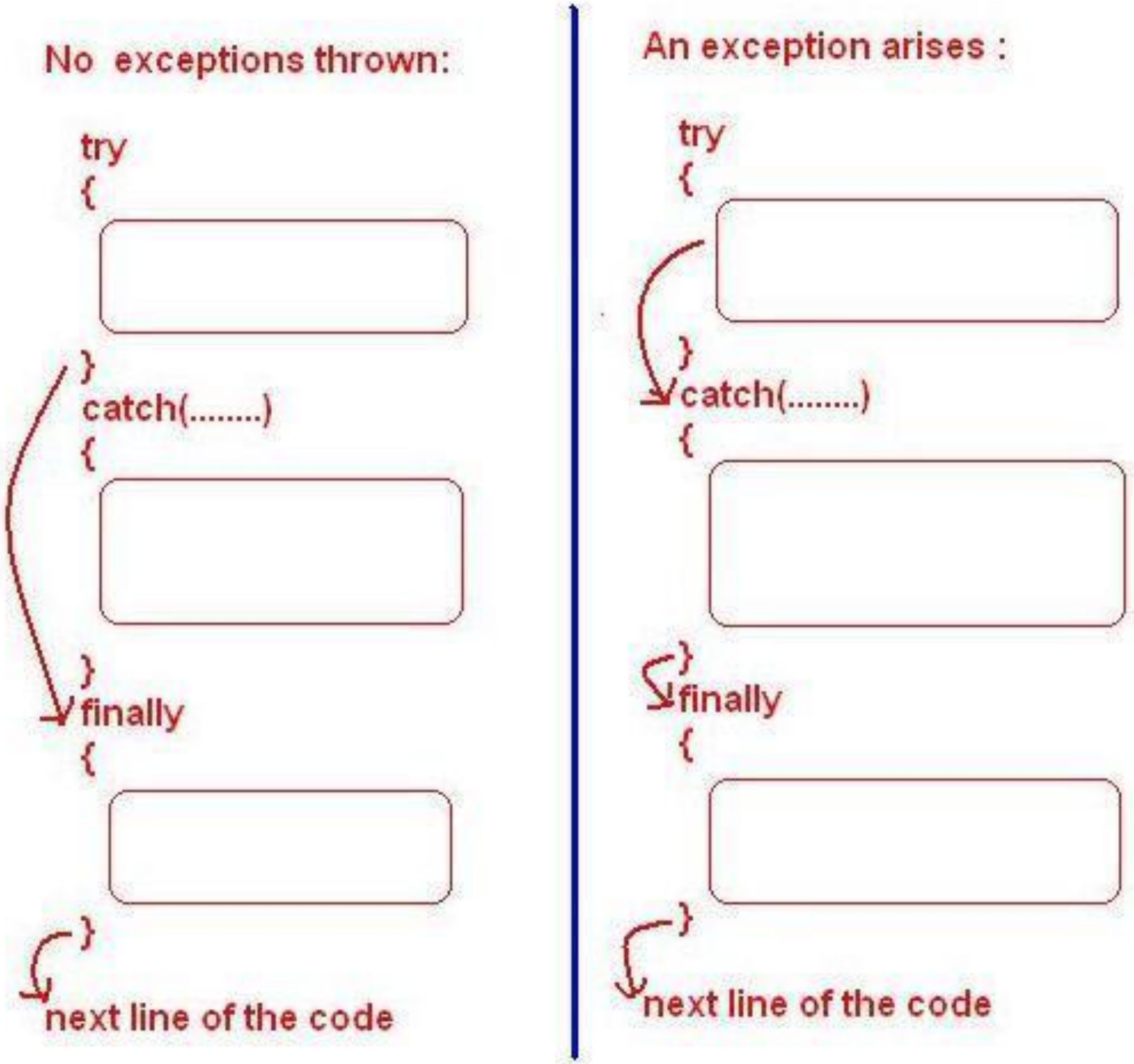
```
try {
    //open file
    //read file
}
catch(Exception e) {
    //handle exception while reading the file
}
finally {
    //close the file
}
```

PDP
IT-ACADEMY

Only try block is mandatory

```
try {

} catch(Exception e) {

}
```

```
try {

} finally {

}
```


No exceptions thrown:
try
{
}
catch(........)
{
}
finally
{
}
next line of the code
An exception arises :
try
{
}
catch(........)
{
}
finally
{
}
next line of the code

PDP
IT-ACADEMY

Java throw keyword

- Syntax
- Handling of unchecked exceptions
- Handling of checked exceptions
- Exception propagation

Syntax

```
public void method() {
        //throwing an exception
        throw new SomeException("message");
}
```

Handling of unchecked exceptions

```
public class JavaExample {
    public static void main(String[] args)    {
        method();
    }
    public static void method( ) {
        throw new NullPointerException();
    }
}
```

Handling of checked exceptions

```java
public class JavaExample {
    public static void main(String[] args)  {
        try {
            method();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static void method( ) throws FileNotFoundException {
        throw new FileNotFoundException();
    }
}
```

Exception propagation

- Exception ni ushlab olib ishlov berilmasa methoddan chaqirilgan methodga o’tadi. Masalan: a method b methodni, b method c methodni, c method d methodni chaqirgan bo’lsa, exception a dan b ga, b dan c ga, c dan d methodga o’tishi mumkin, agarda birorta method da ushlab ishlov berilmasa

- Exceptionga ishlov beruvchini izlash exception sodir bo’lgan joydan boshlanadi.

- Agar ishlov beruvchi (*catch*) topilsa u orqali ishlov beriladi, topilmasa JVM gacha yetib boradi.

Java throws keyword

- **Syntax**
- **Can throw checked and unchecked exceptions**

Syntax

```
public static void method( ) throws FileNotFoundException, ConnectionException {
    //code
}
```

PDP
IT-ACADEMY

Can throw checked and unchecked exceptions

```
public class JavaExample {

    public static void main(String[] args)     {

        try    {
            //Can skip handling of NullPointerException (unchecked exception)
            method();
        }
        catch (FileNotFoundException e)   {
            e.printStackTrace();
        }
    }

    public static void method( ) throws NullPointerException, FileNotFoundException {
        //code
    }
}
```

try-with-resources

```
try (BufferedReader br = new BufferedReader(new FileReader("C:/temp/test.txt"))) {
    String sCurrentLine;
    while ((sCurrentLine = br.readLine()) != null) {
        System.out.println(sCurrentLine);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Create new uncheked exception

```
public static void main(String[] args) {
 throw new MyException();
}
public class MyException extends NumberFormatException {
    public MyException() {
        super("Exeption info");
    }
}
```

Create new cheked exception

```
public static void main(String[] args) {
    try {
        throw new MyException();
    } catch (MyException e) {
        e.printStackTrace();
    }
}


public class MyException extends Exception {

    public MyException() {

        super("Exeption info");

    }

}
```

PDP
IT–ACADEMY

E’TIBORINGIZ UCHUN RAXMAT