TechWeeks

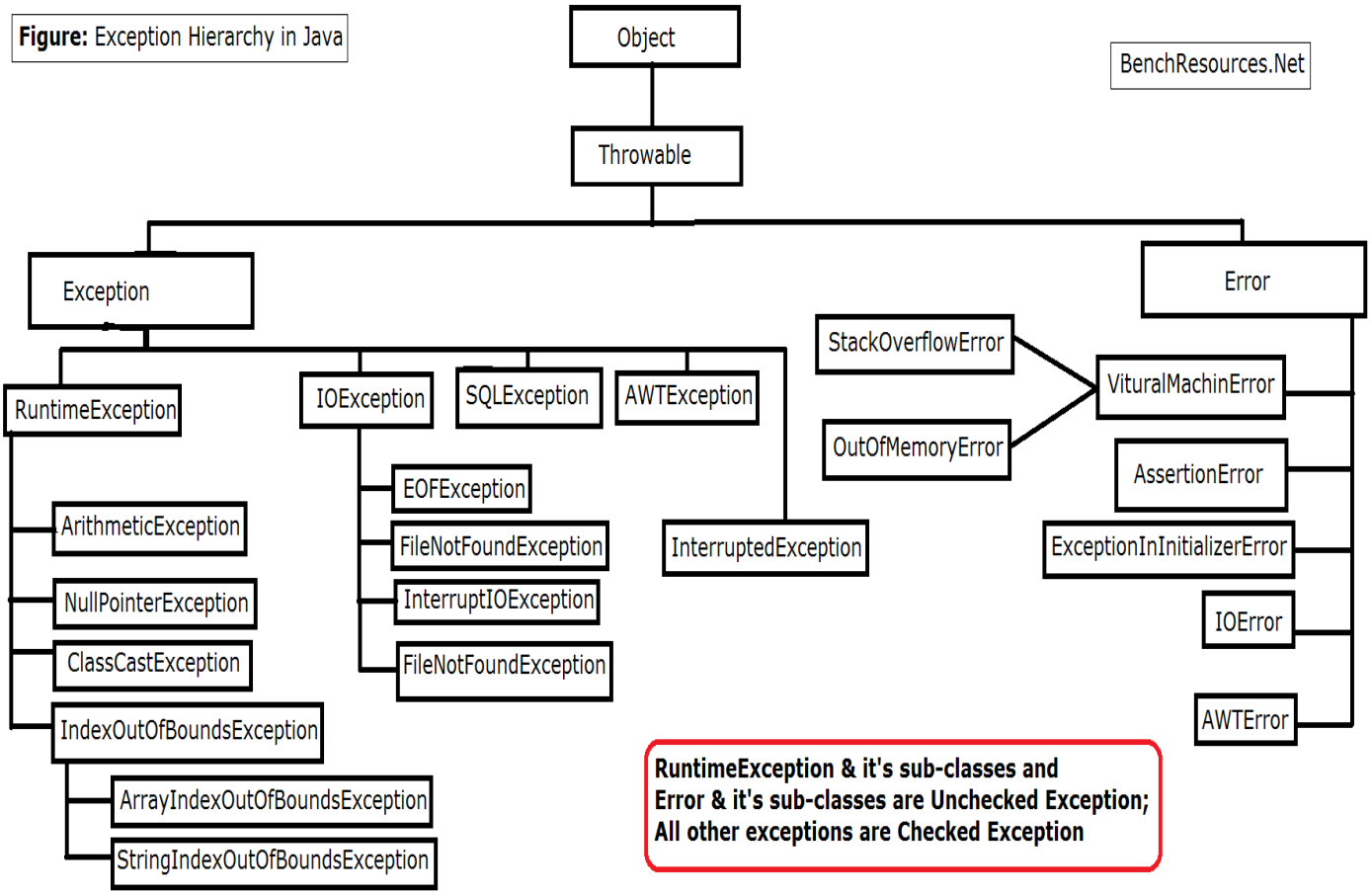
Day8: Topics for today!!!!

Exception Handling and Debugging

- **Debugging** is the process of detecting and removing of existing and potential errors (also called as 'bugs') in a software code that can cause it to behave unexpectedly or crash.
- It is the most important task to understand becoz as a programmer we all run into some bugs and error so we need to know this topic too.
- Debugging allows you to run a program interactively while watching the source code and the variables during the execution.
- But nowadays we have IDE which can do most of our debugging for us so to understand debugging we need to know About **EXCEPTION HANDLING.**

➢ What is a Exception ?

- An **exception** (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the **normal flow** of the program is **disrupted** and the program/Application **terminates abnormally**, which is not recommended, therefore, these exceptions are to be handled.
- An exception can occur for many different reasons. Following are some scenarios where an exception occurs.
- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the **JVM** has run out of memory.
- There are two types of Exception handling :
  - ➢ Runtime Exception Handling .
  - ➢ Compile time Exception Handling.



1. Learn more about exceptions from below links:
   - https://www.javatpoint.com/exception-handling-in-java
2. Exception Handling can be done using :
   - Try – catch
   - Try – finally

➢ Try – catch : Syntax

```
try{
//code that may throw an exception
}catch(Exception_class_Name ref){}
```

➢ Try – finally : Syntax

```
try{
//code that may throw an exception
}finally{}
```

- Whenever you encounter any error you can always Try to solve that error by handling the exception during the Runtime or Compile time
- This means that you need to determined the error, create the conditional statement to avoid the error and if that statement fails that's when Catch block comes into picture
- Catch block helps that error to locate and you can always print the error onto the screen (command line)
- Above the hierarchy of Exceptions and also the link to understand better:
- https://airbrake.io/blog/java-exception-handling/the-java-exception-class-hierarchy

- Other then that finally block is the one which is accessed even if there is a error or not ie if your code doesn't have a error still the finally block will run

- Exception : 2 types
- Pre defined
- User defined

3. Pre-defined are the exception we use to maintain the error checking within our code
4. User-defined are those which are created by us and raised during the run-time
5. They are raised by the keyword extends :
6. https://www.javatpoint.com/custom-exception
7. Other then try catch finally you can also throw the exception as per your choices
8. https://www.javatpoint.com/throw-keyword

1.Example codes on try – catch – throw – finally

1. Try-catch :

```
public class TryCatchExample2 {
   public static void main(String[] args) {
      try
      {
      int data=50/0; //may throw exception
      }
         //handling the exception
      catch(ArithmeticException e)
      {
         System.out.println(e);
      }
      System.out.println("rest of the code");
   }
}
```

2. Try-catch-finally:

```
class TestFinallyBlock{
 public static void main(String args[]){
 try{
  int data=25/5;
  System.out.println(data);
 }
 catch(NullPointerException e){System.out.println(e);}
 finally{System.out.println("finally block is always executed");}
 System.out.println("rest of the code...");
 }
}
```

3. Throw exception:

```
public class TestThrow1{
  static void validate(int age){
    if(age<18)
     throw new ArithmeticException("not valid");
    else
     System.out.println("welcome to vote");
  }
  public static void main(String args[]){
     validate(13);
     System.out.println("rest of the code...");
  }
}
```

Assignment for day7

1. Write the code for basic subtraction between 2 numbers, and Do all the validations in the code using try-catch-finally.

2. Write a factorial of a number and do validation in the code using Throw block.

3. Find the output of code:

```
public class Test
{
   public static void main(String[] args)
   {
      try
      {
         System.out.printf("1");
         int data = 5 / 0;
      }
      catch(ArithmeticException e)
      {
         System.out.printf("2");
         System.exit(0);
      }
      finally
      {
         System.out.printf("3");
      }
      System.out.printf("4");
   }
}
```

4. Find the output of code:

```
import java.io.EOFException;
import java.io.IOException;

public class Test
{
   public static void main(String[] args)
   {
      try
      {
         System.out.printf("1");
         int value = 10 / 0;
         throw new IOException();
      }
      catch(EOFException e)
      {
         System.out.printf("2");
      }
      catch(ArithmeticException e)
      {
         System.out.printf("3");
      }
      catch(NullPointerException e)
      {
         System.out.printf("4");
      }
      catch(IOException e)
      {
         System.out.printf("5");
      }
      catch(Exception e)
      {
         System.out.printf("6");
      }
   }
}
```