EXCEPTIONS

- Exception is an error event which happens during the execution of a program. It disrupts the normal flow of program.
- Database server down, hardware failure, network connection failure

Exception Handling

- Overcoming problems during run-time or exceptions.
- Java being OOP language wherever an error occurs while executing a statement, creates an exception object and then the normal flow of program halts and JRE tries to find someone who can handle the raised exception
- Exception object contains a lot of debugging information such as method hierarchy, line number where the exception occurred, type of exception etc

```java
package com.dev.exception;

public class ExcepionExample {

    public static void main(String[] args) {
        t();
    }
    public static void p() {
        StringBuffer sb = new StringBuffer(-1);
    }

    public static void t() {
        p();
    }
}
```

```
Exception in thread "main" java.lang.NegativeArraySizeException
at java.lang.AbstractStringBuilder.<init>(AbstractStringBuilder.java:68)
at java.lang.StringBuffer.<init>(StringBuffer.java:128)
at com.dev.exception.ExcepionExample.p(ExcepionExample.java:21)
at com.dev.exception.ExcepionExample.t(ExcepionExample.java:25)
at com.dev.exception.ExcepionExample.main(ExcepionExample.java:14)
```

Method Hierarchy

- When the exception occurs in a method, the process of creating an exception object and handling it to run-time environment is called throwing an exception.
- Once runtime receives the exception object, it tries to find the handler for the exception. Exception handler is a block of code that can process the exception object.
- Logic :Starting the search in the method where error occurred. If not found then it is passed to calling method and so on.
- If any exception is found, exception object is passed to handler to process it. The handler is said o be exception handler.
- Exception cannot solve compile time errors.
- Differentiate exceptions and errors:

  Errors are solved using programming skills, errors occur during compile time

  Exceptions are solved using exception handler, occurs during run-time

try-catch – used for exception handling our code. Try can have multiple catch bloc and we can have nested try-catch. Catch block should take parameter that should be of type exception. If exception parameter need to be included in catch,then should be written in last catch block .[Try can have arguments.(resources from JDBC)]

```java
package com.dev.exception;

public class ExceptionHandling {

    public static void main(String[] args) {
        try {
            p();
        }catch(Exception e) {
            System.out.println("Exception Handling");
        }
        finally {
            System.out.println("Final Block");
        }
        System.out.println("outside try catch");
    }

    public static void p() {
        StringBuffer sb = new StringBuffer(-1);
    }

}
```

```
<terminated> ExceptionHa
Exception Handling
Final Block
outside try catch
```

```java
package com.dev.exception;

public class ExceptionHandling {

    public static void main(String[] args) {
        try {
            p();
        }catch(Exception e) {

        }
    }

    public static void p() {
        StringBuffer sb = new StringBuffer(-1);
    }

}
```

```
<terminated> ExceptionHandli
```

finally → Code written here will be executed even if there is try-catch or any exceptions. Finally can have try catch

```java
package com.dev.exception;

public class ExceptionHandling {

    public static void main(String[] args) {
        try {
            p();
        }/*catch(Exception e) {
            System.out.println("Exception Handling");
        }*/
        finally {
            System.out.println("Final Block");
        }
        System.out.println("outside try catch");
    }

    public static void p() {
        StringBuffer sb = new StringBuffer(-1);
    }

}
```

```
<terminated> Except
Final Block
Exception in thr
        at java.
        at java.
        at com.c
        at com.c
```

throw – Exception defined or thrown by developer

```java
package com.dev.exception;

public class CustomException extends Exception {
    public CustomException() {
        System.out.println("Custom Exception");
    }

    public  CustomException(int i) {
        System.out.println("Custom Exception for integer");
    }

    public  CustomException(String i) {
        System.out.println("Custom Exception for String");
    }

}
```

```java
package com.dev.exception;

public class ExceptionHandling extends CustomException{

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

        s();
        try {
            divide(10,0);
        } catch (Exception e) {
            throw new CustomException();
        }
    }

    public static int divide(int i,int j) {
        int res =i/j;
        System.out.println(res);
        return 1;

    }

    public static void s() {
        try {
            StringBuffer sb = new StringBuffer(-1);
        } catch (Exception e) {
            new CustomException().printStackTrace();
        }
    }
```

```
Custom Exception
com.dev.exception.CustomException
        at com.dev.exception.ExceptionHandling.s(ExceptionHandling.java:26)
        at com.dev.exception.ExceptionHandling.main(ExceptionHandling.java:7)
Custom Exception
Exception in thread "main" com.dev.exception.CustomException
        at com.dev.exception.ExceptionHandling.main(ExceptionHandling.java:11)
```

- **throws** –where exception is pre-defined. It is used when calling method is throwing an exception. It tells that a method might thrown an exception
- When a method is declared as throws then the method which calls this method should also be declared as throws.
- Can provide more than one exception at once.

```java
package com.dev.exception;

public class ExceptionHandling {

    public static void main(String[] args) throws Exception {
        p();
        System.out.println("code after execuion");
    }

    public static void p() throws Exception {
        StringBuffer sb = new StringBuffer(-1);
    }

}
```

```
<terminated> ExceptionHa
Exception in thread
        at java.lang
        at java.lang
        at com.dev.e
        at com.dev.e
```

- Java exceptions are hierarchical and inheritance is used to categorize different types of exception.
- **throwable** is a parent class of java exceptions hierarchy and it has two child objects – Error and Exceptions.
- Errors : Errors are exceptional scenarios that are out of scope of application and its not possible to anticipate and recover from them.
- Ex: Hardware failure,JVM crash or out of memory error.

CHECKED EXCEPTIONS	UNCHECKED EXCEPTIONS
Exception scenarios which can be anticipated are **checked** Exception.	
Provide errors at compilation time	warnings and errors during run-time.
Should be caught and provide useful information to user(will be written in catch).	
Try and catch for catch exceptions	try and finally in **unchecked** Exception.