Exception

Objectives

- Exception Introduction
- Exception Types
- Handling Mechanism
- Custom Exception

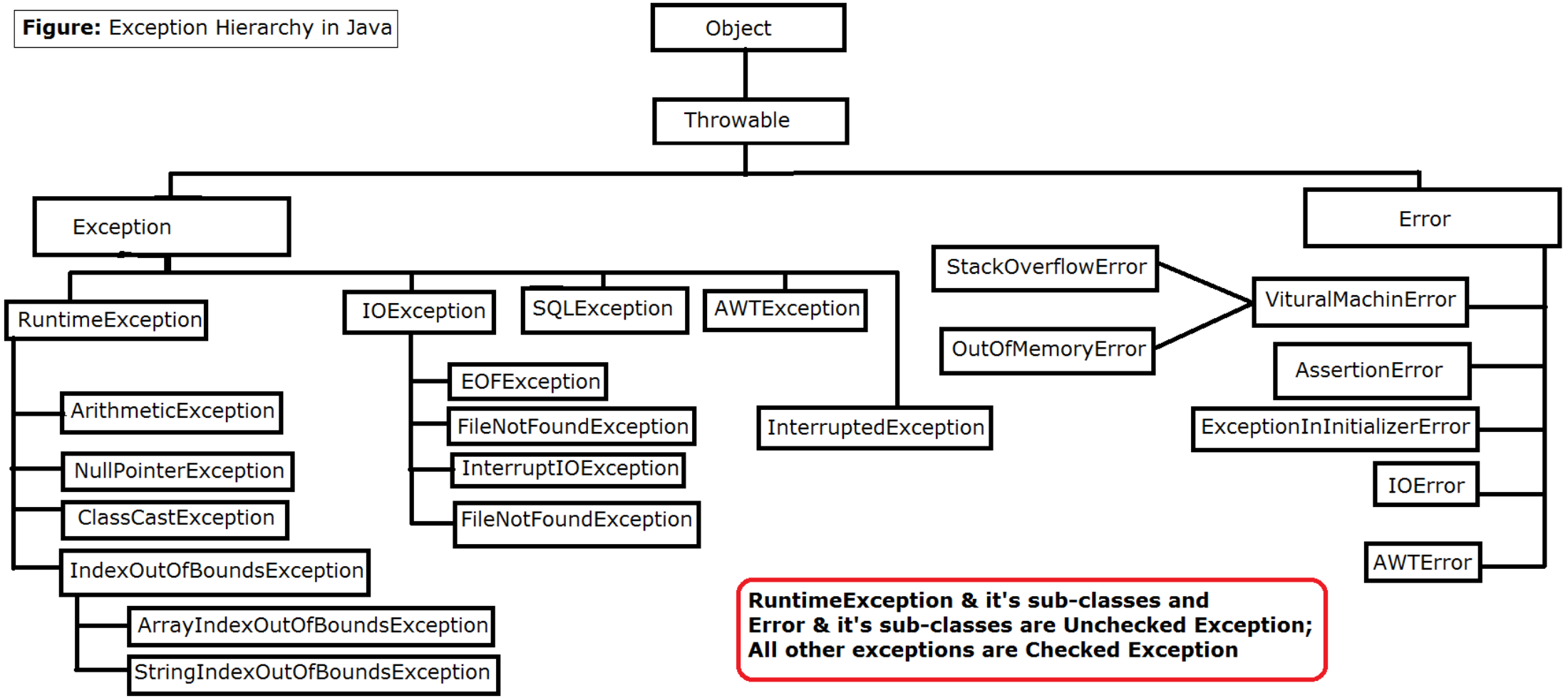
Exception Introduction

Exception

- An exception is an unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program’s instructions.

Exception Contd…

Figure: Exception Hierarchy in Java



Exception Types Contd…

Exception Types

Exception Types Contd…

1) Error

- Error is irrecoverable
- e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

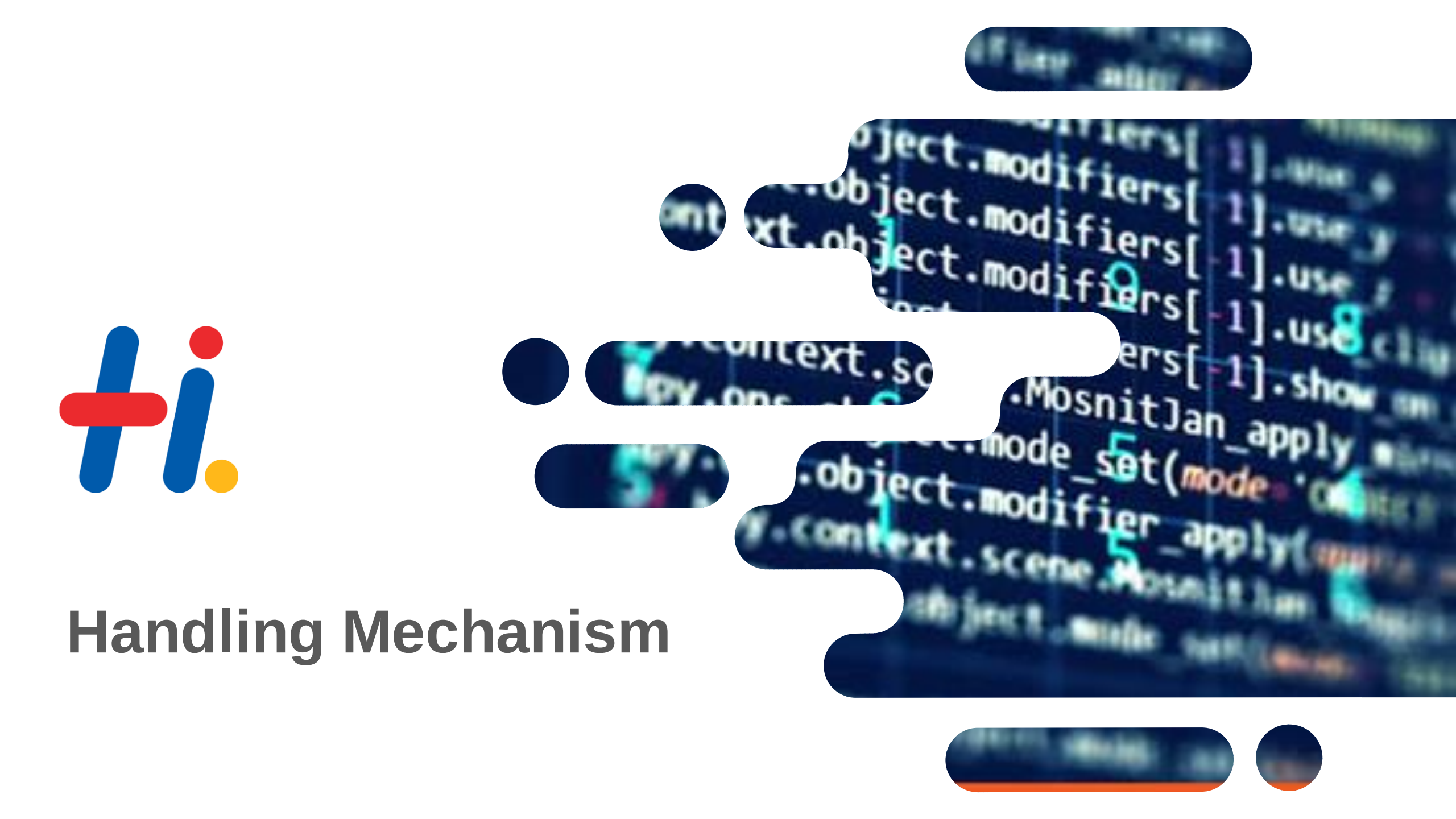
2) Checked Exception

- The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions
- e.g. IOException, SQLException etc.
- Checked exceptions are checked at compile-time.

Exception Types Contd…

3) Unchecked Exception

- The classes which inherit RuntimeException are known as unchecked exceptions
- e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.
- Unchecked exceptions are not checked at compile-time, but they are checked at runtime.
- System-generated exceptions are automatically thrown by the Java run-time system.



Handling Mechanism

Exception Handling Mechanism

- ✓Try
- ✓Catch
- ✓Throw
- ✓Throws
- ✓Finally

Handling Mechanism Contd…

Try & Catch:

- If an exception occurs within the try block, that exception is handled by the exception handler associated with it.
- Each catch block is a exception handler that handles the exception of the type indicated by its argument.
- Program statements that can raise exceptions are contained within a try block. If an exception occurs within the try block, it is thrown.

```
try{
  int data=25/5;
  System.out.println(data);
 }
 catch(NullPointerException e){System.out.println(e);}
```


Handling Mechanism Contd…

Throw:

- **Throw** keyword is used inside a function. It is used when it is required to throw an Exception logically.
- It is used to throw an exception explicitly.
- It can throw only one exception at a time.
- To manually throw an exception, use the keyword throw.

```
if(age<18)
    throw new ArithmeticException("not valid");
```


Handling Mechanism Contd…

Throws:

- **Throws** keyword is in the function signature. It is used when the function has some statements that can lead to some exceptions.
- It is used to declare multiple exceptions, separated by comma.
- Any exception that is thrown out of a method must be specified as a throws clause.

```
void method()throws IOException{
  throw new IOException("device error");
 }
```


Handling Mechanism Contd…

Finally:

- Finally block is used to execute important code such as closing connection, stream etc.
- It is always executed whether exception is handled or not.
- It follows try or catch block.

```
finally{
System.out.println("finally block is always executed");
}
```


Handling Mechanism Contd…

Do you Know:

- We can’t have catch or finally clause without a try statement.
- A try statement should have either catch block or finally block, it can have both blocks.
- We can’t write any code between try-catch-finally block.
- We can have multiple catch blocks with a single try statement.
- try-catch blocks can be nested similar to if-else statements.
- We can have only one finally block with a try-catch statement.

Exception Example

```
try{
        int a=10,b=2,c,sum;
        int eid[]=new int[]{22,33,44};
        Scanner s=new Scanner(System.in);
        String d=s.next();
        int add=Integer.parseInt(d);
           sum=a+add;
           System.out.println("sum:"+sum);
        c=a/b;
        System.out.println(c);
        System.out.println(eid[21]);
        }
```

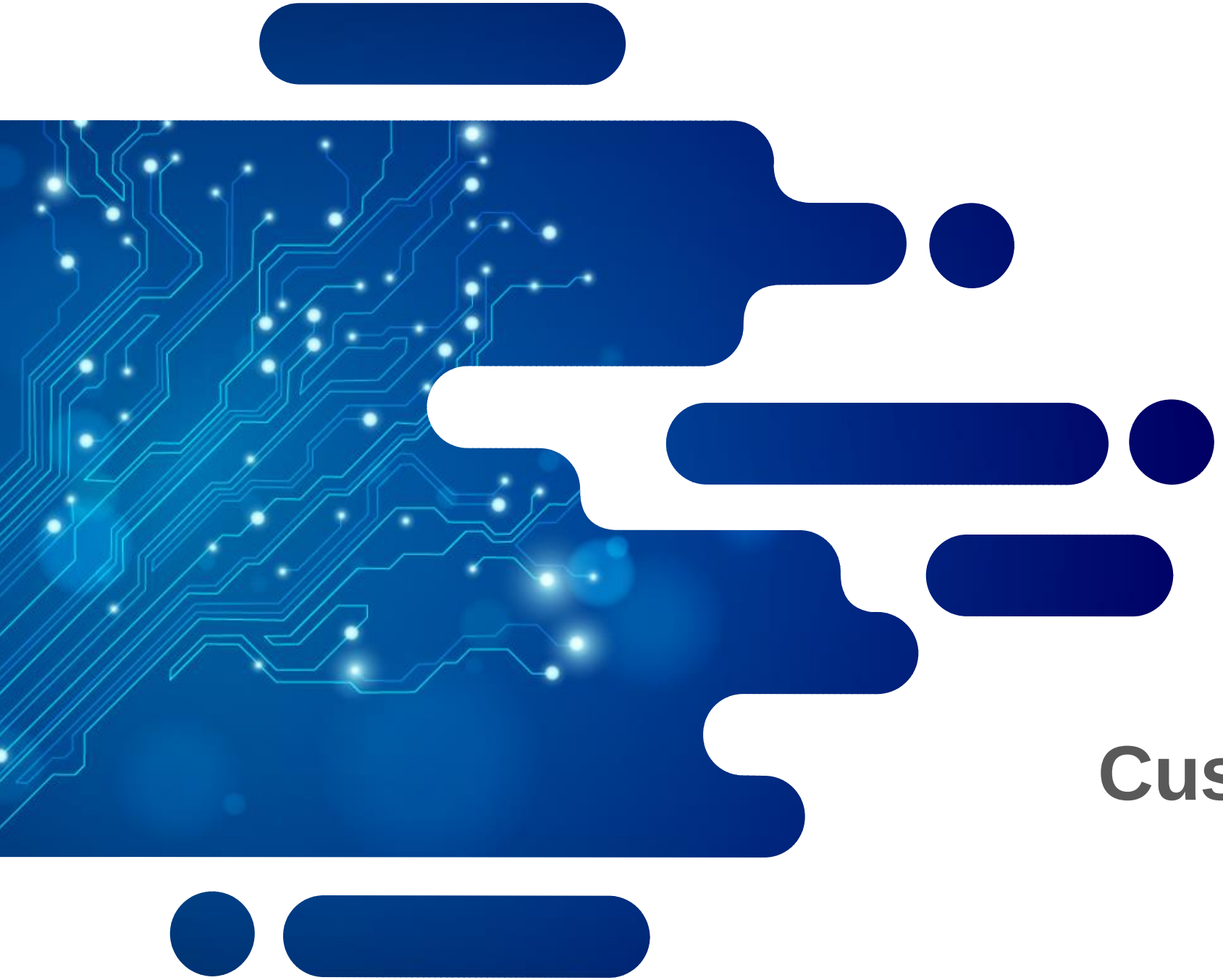

Exception Example

```
catch(ArithmeticException e){
        System.out.println("Enter only integers   "+e);
    }
    catch(ArrayIndexOutOfBoundsException e1){
        System.out.println("Index value is not correct    "+e1);
    }
    catch(NumberFormatException  e3){
        System.out.println("Enter only integers for sum   "+e3);
    }finally{
        System.out.println("I am finally");
    }
}
```


Java 7 Automatic Resource Management and Catch block improvements

- In java 7, one of the improvement was try-with-resources where a resource is created in the try statement itself and used inside the try-catch block.
- When the execution comes out of try-catch block, runtime environment automatically close these resources.

```
static String readFirstLineFromFile(String path) throws IOException {
    try (BufferedReader br =
                   new BufferedReader(new FileReader(path))) {
        return br.readLine();
    }
}
```



Custom Exception

Custom Exception Contd…

- To create our own custom exception classes to notify the caller about specific type of exception with appropriate message and any custom fields to introduce for tracking, such as error codes.

Custom Exception Contd…

```
class CheckedException extends Throwable{
   public CheckedException(){
       super();
       }
}
class UserException{
    public static void checkSalary(int salary) throws CheckedException{
       int amount=0;
       if(salary>10000)
       //raise Exception
       throw new CheckedException();
       else{
          amount=salary+5000;
       System.out.println("Ur total salary is:"+amount);
    }
}
```


Custom Exception

```
public static void main(String[] args) {
    try{
    checkSalary(1300);
    }catch(CheckedException ce){
        System.out.println("U r not eligible for bonus ");
    }
}
}
```

Exception – Practice Question

Exception -
Practice question

thank you

www.hexaware.com