EXCEPTIONS

Java uses *exceptions* to handle errors and other exceptional events .
Exceptions are always runtime errors and cannot occur at compile time

Some examples

ClassNotFoundException	(Class definition not found during execution of program)
ArrayIndexOutOfBoundsException	(Index not within range)
StringIndexOutOfBoundsException	(Index not within range of the string)

CATEGORIES OF EXCEPTIONS

Checked Exceptions

All the checked exceptions must be handled else compile-error results

Examples of checked exceptions

IOException
FileNotFoundException

Unchecked Exceptions

Unchecked exceptions are the exceptions that may or may not be handled
All the unchecked exceptions are sub-classes of RuntimeException

Examples of unchecked exceptions

ArithmeticException	(Results when an attempt is made to divide by zero)
NumberFormatException	(Invalid number format)
NullPointerException	(Trying to call methods or access fields through a reference pointing to null)

HANDLING EXCEPTIONS

The most general way of handling exceptions is shown below.

try

```
{
Statement1;
Statement2;
……
}
catch(Exception1 e1)
{
        //handle Exception1
}
catch(Exception2 e2)
{
        //handle Exception2
}
…..
catch(ExceptionN eN)
{
        //handle ExceptionN
}
finally
{
        ///this code will be executed irrespective of whether an exception occurs or not
}
```

All the statements that might result in errors can be enclosed inside the try block .
A try block must be followed by one or more catch blocks and an optional finally block .
The ith catch statement will catch the exception of type ExceptionI
eI refers to the object will contains all the details of the error

Case 1 :- No error
If there is no error inside the try block none of the catch blocks are executed

Case 2 :- An error occurs at statement i
If there is an error at statement i in try block , all the statements after statement i in the try block will be skipped and control will be passed to the matching catch statement

Case 3 :- An error occurs at statement i and there is no matching catch statement

If there is an error at statement i in try block , all the statements after statement i in the try block will be skipped and control will be passed to the callee . If the exception is not caught anywhere in the program , the program will crash and the stack trace will be printed which gives all the details about the exception

THROWING AN EXCEPTION

An object of type Exception can be thrown using the throw clause .

Example

```
try
{
…..
throw new ArithmeticException(“Divide by zero”);
….
}
catch(ArithmeticException e1)
{
System.err.println(e1);
}
finally
{
System.out.println(“This statement is always executed”);
}
```

In the above code snippet , an object of type ArithmeticException with the message “Divide by zero” is created and thrown in the try block .
This exception will be caught by the exception handler(catch statement) which catches the exceptions of type ArithmeticException .
The object which contains the details of the error is assigned to e1 automatically
When this object is printed using System.err.println , the toString method of that object is called .
The finally statement is executed after the exception handler

The output of the program is ,

ArithmeticException:Divide by zero
This statement is always executed