URBAN EDGE

# Chapter 14 - Errors & Exceptions

No matter how smart we are, errors are our constant companions. With practice, we keep getting better at finding & correcting them.

There are three types of errors in Java.

1. Syntax errors
2. Logical errors
3. Runtime errors → Also called Exceptions!

## Syntax Errors

When compiler finds something wrong with our program, it throws a syntax error.

```
int a = 9          → No semicolon, syntax error!
a = a+3;

d = 4;  → Variable not declared, Syntax error!
```

## Logical errors

A logical error or a bug occurs when a program compiles and runs but does the wrong thing.

→ message delivered wrongly
→ Wrong time of chats being displayed
→ incorrect redirects!

## Runtime Errors

Java may sometimes encounter an error while the program is running. These are also called exceptions!

URBAN EDGE

These are encountered due to circumstances like bad input and (or) resource constraints.
Ex: user supplies '5'+8 to a program which adds 2 numbers.

Syntax errors and logical errors are encountered by the programmer where as Runtime errors are encountered by the users.

Exceptions in Java
An Exception is an event that occurs when a program is executed disrupting the normal flow of instructions.
There are mainly two types of exceptions in Java:
1) Checked Exception → Compile time exceptions (Handled by Compiler)
2) Unchecked Exception → Runtime exceptions

Commonly occuring Exceptions
Following are few commonly occuring exceptions in Java:
1) Null Pointer Exception
2) Arithmetic Exception
3) Array Index Out of Bound Exception
4) Illegal Argument Exception
5) Number format Exception

try-catch block in Java
In Java, exceptions are managed using try-catch blocks
Syntax:

```
try {
    // Code to try }
  catch (Exception e) {
    // Code if exception
    }
```

## Handling specific Exceptions

In Java, we can handle specific exceptions by typing multiple catch blocks.

```
try {
    // Code
}
Catch (IOException e) {          → Handles all Exceptions of type IOException
    // Code
}
Catch (ArithmeticException e) {  → Handles all Exceptions of type ArithmeticException
    // Code
}
Catch (Exception e) {            → Handles all other Exceptions
    // Code
}
```

## Nested try-catch

We can nest multiple try-catch blocks as follows:

```
try {
    try {
    }
    Catch (Ex. e) {
    }
}
Catch (Ex. e) {                  ⟹ Nested try-catch blocks
}
```

Similarly, we can further nest try catch blocks inside the nested try catch blocks.

URBAN EDGE

Quick Quiz: Write a Java program that allows you to keep accessing an array until a valid index is given by the user.

Exception class in Java

We can write our custom Exceptions using Exception class in Java.

```
public class MyException extends Exception {
        //overridden methods
        }
```

The Exception class has following important methods:

1. String toString() → executed when sout(e) is ran
2. Void printStackTrace() → prints stack trace
3. String getMessage() → prints the Exception message

The Throw Keyword

The throw keyword is used to throw an exception explicitly by the programmer

```
if (b==0) {
    throw new ArithmeticException("Div by 0");
    }
else {
 return a/b;
    }
```

In a similar manner, we can throw user defined exceptions:

```
throw new MyException("Exception thrown");
```

The throws exception

The Java throws keyword is used to declare an exception. This gives an information to the programmer that there might be an exception so, its better to be parepared with a try catch block!

```
public void calculate (int a, int b) throws IOException {
        // code
    }
```

Java finally block

finally block contains the code which is always executed whether the exception is handled or not.

It is used to execute code containing instructions to release the system resources, close a connection etc.