CHAPTER 6: UNDERSTANDING ERRORS

6.1 Preamble

By this point you should have some Java coding experience and must have come across a lot of errors. Some are easy to figure out, some are difficult. In this chapter we shall learn about some common errors and how to prevent these. After completing this chapter, you will no longer have to spend hours trying to figure out what error occurred. Errors in Java can be categorized as follows:

- Syntax Errors
- Logical Errors
- Runtime Errors or Exceptions

6.2 Syntax Errors

Syntax error occurs while we make mistakes writing syntaxes. It is usually caused by the most trivial mistakes or lack of attention. These are the easiest to spot and Java compilers find these for you including the line numbers. This is why they are also known as compile-time errors. The program cannot generate a ".class" file. Some common syntax errors are shown below:

Reason	Example	Error Shown
Semicolon or brackets missing	System.out.println("Hi")	';' expected
Not closing first brackets	System.out.println("Hi!";	')' expected

Not closing curly braces	class HelloWorld {
	    public static void main(String[] args) {
	        x=1; }
		reached end of file while parsing

Misspelled keywords	public staic void main(String[] args) {	<identifier> expected
Wrong-Cased keywords	Public static void main(String[] args) {	<identifier> expected
Improper variable names	public= x; //Reserved keyword	illegal start of expression
	1name= 10; //variable name starts with a digit	not a statement
Missing double quotes in Strings	System.out.println("Hi);	unclosed string literal
Using undeclared variables	print(x); //x not declared	cannot find symbol
Missing or excessive operators	y = 3 + * 5; //excessive operators	illegal start of expression

6.3 Logical Errors

Logical errors occur when the syntaxes are correct, successfully compiles, detects no errors during runtime but the code generates wrong output. Despite logical errors, your code will run, the program will generate a ".class" file, but the output will be wrong.

Task	Code	Problem

Add two integer numbers and print the summation.	x=1;
	y=5;
	System.out.println(x*y);
		The output was supposed to be 6, but we got 5. Here the code is syntactically correct but contains logical error.

There is no hard and fast rule for preventing logical errors. You have to crosscheck all the logics if the output does not match.

6.4 Runtime Errors or Exceptions

Runtime errors occur when the program successfully compiles without any error and creates a ".class" file. However, the program does not execute properly. These errors are detected at runtime or at the time of execution of the program instead of compile time. These runtime errors are called exceptions and they terminate the program abnormally, giving an error statement. A runtime error can happen when:

Reason	Example	Error Shown
Division by Zero	System.out.println(5/0);	java.lang.ArithmeticException: / by zero
Input Mismatch	int x = myObj.nextInt(); //Here an int is expected but if you enter a String instead, this will lead to an exception	java.util.InputMismatchException

Null Pointer	String s = null;
	System.out.println(s.length());
	// null has no length
		java.lang.NullPointerException

Wrong	int x = Integer.parseInt("Hi"); //Converting a string with improper format into a numeric value	java.lang.NumberFormatException : For input string: "Hi"

We shall learn more about Exceptions and how to handle them later down the line.

6.5 Worksheet

A. Check the following code carefully and find all the possible errors and then rewrite it correctly. The output of this code should be:
false
15

Given Code Lines	**Error Present or Not** (If present specify which error)	**Rewritten Code**
class A {		
public static void man(String[] args) {		
int x= 5/0;		
int y= 7		
x-==1;		
1y+=1;		
System.out.println(y%2!=0;		
int m = Integer.valueOf("Hi");		
System.out.println(m);		
}		