Java Development.

WEEK 2

Naralasetti Umesh Surya Kiran

- Data Types
- Control Flow Structures
- Exception handling
- File Handling

Data Types in Java

1.int:

- •Represents a 32-bit signed integer.
- •Range: -2^31 to (2^31 - 1).
- •Example: **int age = 25;**

2.float:

- •Represents a 32-bit IEEE 754 floating-point.
- •Suitable for decimal numbers with moderate precision.
- •Example: **float price = 49.99f;**

3.double:

- •Represents a 64-bit IEEE 754 floating-point.
- •Higher precision compared to float.
- •Example: **double distance = 1234.5678;**

4.String:

- •Represents a sequence of characters.
- •Immutable, meaning once created, the content cannot be changed.
- •Example: **String greeting = "Hello, Java!";**

5.char:

- •Represents a single 16-bit Unicode character.
- •Enclosed in single quotes (' ').
- •Example: **char grade = 'A';**

6.boolean:

- •Represents a true or false value.
- •Used for logical operations and conditions.
- •Example: **boolean isJavaFun = true;**

Control Flow Structures

If-else:

- Conditionally executes a block of code based on a boolean expression.
- Allows handling both true and false cases.

```
int number = 7;
if (number % 2 == 0) {
    System.out.println("Even");
} else {
    System.out.println("Odd");
}
```

Switch:

- Provides multi-branch selection based on the value of an expression.
- Offers a cleaner alternative to nested if-else statements for multiple conditions.

```
int day = 3;
switch (day) {
    case 1: System.out.println("Monday");
break;
    case 2: System.out.println("Tuesday");
break;
    default: System.out.println("Invalid day");
}
```

For loop:

- Iterates a specific number of times using a control variable.
- Consists of initialization, condition, and iteration expressions.

```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

Control Flow Structures

For Each Loop:

- **Concise Iteration:** Simplifies array and collection loops without indexing.
- **Read-Only Access:** Ensures safe traversal, preventing unintended modifications.

```
int[] numbers = {1, 2, 3, 4, 5};
for (int num : numbers) {
    System.out.println(num);
}
```

While loop:

- Repeats a block of code while a specified condition is true.
- Suitable when the number of iterations is unknown beforehand.

```
int count = 0;

while (count < 5) {
    System.out.println("Count: " + count);
    count++;
}
```

Exception Handling in java.

- There are two types of errors they are 1) Run Time errors 2) Compile time errors.
- Compile time errors are like Syntax mistake, Methods are not defined and etc.

Compile time error:

- Where as Run time errors are Syntactically correct but we will not get required out put for all inputs.
- Program is running, but not for all inputs.
- Make program unreliable and may damage systems.
- To handle the common possible errors and exceptions, Java offers a class hierarchy.

Exception Handling in java.

Some Run-time errors:

- A User has entered invalid data or invalid data-type.
- Dividing an integer by zero.
- Trying to store a value into a array of an incomplete class or type.
- Null object reference.
- Trying to cast an instance of a class to one of its sub-class.
- A Network connection has been lost in the middle, JVM has run out of memory

Exception Handling in java.

To handle these exceptions :

- *Try-catch*: We will pass some code in try block . If we have a error then code in catch block will be executed.
- *Try-Catch-Finally*: Same as above try block code code will be run, if we have error comes to catch block, Finally block will be run either if error is there or not there.

```
Try{
        //code
    }
Catch{
        //code
     }
```

```
Try{
        //code
    }
Catch{
        //code
     }
Finally{
        //code
     }
```

File Handling in java.

We have 4 types in file handling:

- File
- File Input Stream
- File Output Stream
- Random Access File

Opening a File Object:

```
Import java.io.*;
File MyFile;
MyFile=new File(file_name);
```

```
Import java.io.*;
File MyFile;
MyFile=new
File(path_name,file_name);
```

```
Import java.io.*;
File MyFile;
MyFile=new
File(MyDrtry,file_name);
```

File Handling Example code:

```
public static void main(String[] args) {
// Specify the file path
 String filePath = "example.txt";
// Example: Writing to a File
try {
 // Open the file in append mode (true indicates append mode)
 FileWriter fileWriter = new FileWriter(filePath, true);
 // Write content to the file
fileWriter.write("Hello, File Handling in Java!\n");
// Save changes and close the file
fileWriter.close();
System.out.println("File successfully written and closed."); }
catch (IOException e) { System.err.println("Error writing to the file: " + e.getMessage());
} }
```

Thank You.!