Course Learning Outcomes

- CLO1 : Understand the basic concept of Object Oriented programming in Java. In addition, students will be able to write simple programs
- CLO2 : Learn about variables and several data types. It addition, they will also learn about encapsulation through programming examples
- CLO3 : Learn the concepts of constructors, abstraction, inheritance and polymorphism
- CLO4 : A quick recap of if-else, loops in programming and learning about their implementation in Java
- CLO5 : Recap of the concept of recursion and its implementation in Java
- CLO6 : Learn String handling in Java and Java APIs
- CLO7 : Learn the advance classes for data storage, manipulation and retrieval and java basic data structures
- CLO8 : to consider user inputs and process them. In addition, they will also get accustomed to handling exceptions in programs.
- CLO9 : Learn to work with files: consider input from files, process them and store outputs to different files. They will also be able to work with directories that contain multiple files.
- CLO10 : Learn to use Java to connect to a database system. They will be able to use Java API to connect and execute the query with the database.
- CLO11 : Learn multithreading concepts in java

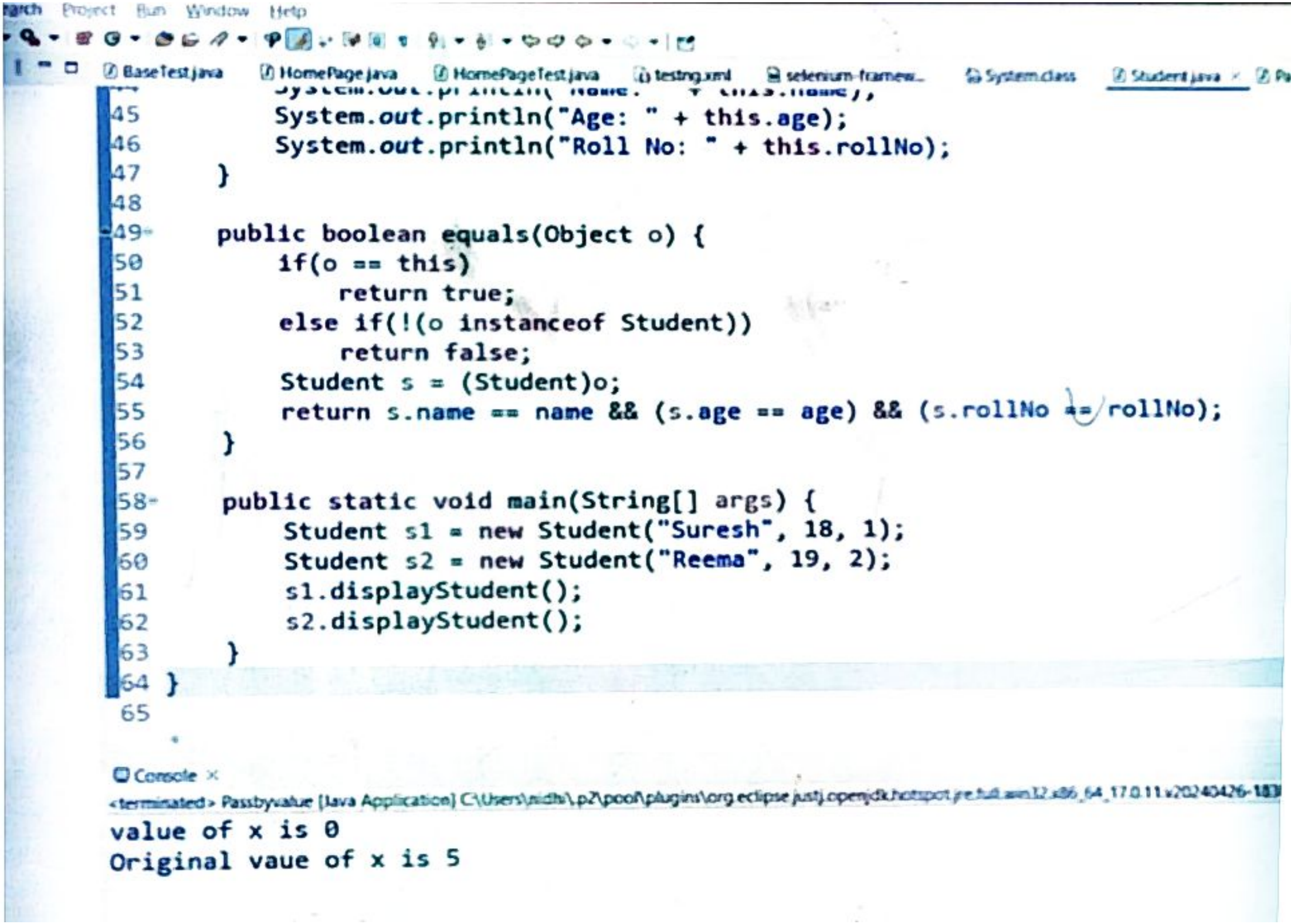
```java
package com.org.sitare.cgo2;

import com.org.sitare.cgo1.Student;

public class Equality {

    public void primitiveEqualty(int x, int y) {
        if(x == y) {
            System.out.println("x is equal to y");
        } else {
            System.out.println("x is not equal to y.");
        }
    }

    public void objRefCompare(Student x, Student y) {
        if(x == y) {
            System.out.println("x is equal to y");
        } else {
            System.out.println("x is not equal to y.");
        }
    }
```

Console

<terminated> Equality [Java Application] C:\Users\...\openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v20240426-1830\jre\

```
x is not equal to y.
x is equal to y
x is not equal to y.
x is equal to y
x is equal to y
```

Writable Smart Insert 7 : 49 : 143


arch Project Run Window Help
BaseTest.java
HomePage.java
HomePageTest.java
testng.xml
selenium-framew...
System.class
Student.java
System.out.println("Name: " + this.name);
System.out.println("Age: " + this.age);
System.out.println("Roll No: " + this.rollNo);
}
public boolean equals(Object o) {
if(o == this)
return true;
else if(!(o instanceof Student))
return false;
Student s = (Student)o;
return s.name == name && (s.age == age) && (s.rollNo == rollNo);
}
public static void main(String[] args) {
Student s1 = new Student("Suresh", 18, 1);
Student s2 = new Student("Reema", 19, 2);
s1.displayStudent();
s2.displayStudent();
}
}
Console
<terminated> Passbyvalue [Java Application] C:\Users\nidhi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.11.v20240426-183
value of x is 0
Original vaue of x is 5

Object Equality

- When we compare primitives with == operator, it compares values of the primitives.
- When you compare two object references, it checks only reference values not the objects.
- If you want to check two objects equality, use equals() method which is method of Object class. Object class is mother of all classes in java.

Packages

- In the Java API, classes are grouped into packages. To use a class in the API, you have to know which package the class is in.

Packages are important for three main reasons.

- First, they help the overall organization of a project or library. Rather than just having one horrendously large pile of classes, they're all grouped into packages for specific kinds of functionality (like GUI, or data structures, or database stuff, etc.)
- Second, packages give you a name scoping, to help prevent collisions. If you and 12 other programmers in your company all decide to make a class with the same name. If you have a class named Set and someone else (including the Java API) has a class named Set, you need some way to tell the JVM which Set class you're trying to use.
- Third, packages provide a level of security, because you can restrict the code you write so that only other classes in the same package can access it.

import the package or type class with full package name

Loops in java

- For Each loop

It is used exclusively to loop through elements in an standard data struc or arraylist.

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Loops in java

- For loop

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

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

There could ne nested loops in your code.

Loops in java

- While loop

```
while (condition) {
  // code block to be executed
}
```

- Do/While loop

```
do {
  // code block to be executed
}
while (condition);
```

break and continue keywords

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```

break and continue keywords

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```

Conditional Statements

- Ternary operator(short hand if else) syntax

```
variable = (condition) ? expressionTrue :  expressionFalse;
```

There could be nested if else statements also.

- Switch statement

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Conditional Statements

ava has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is tru
- Use else to specify a block of code to be executed, if the same condition is f
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

lse syntax

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

lse if syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Java Operators

Difference between (&&, ||) and (&,|)

&&, || : These are called Logical AND and Logical OR operator

&, | : These are called bitwise AND and bitwise OR operator

S.N.	Basis	& Operator	&& Operator
1	**Operator**	It is a bitwise AND operator.	It is a logical AND operator.
2	**Evaluation**	It evaluates both the left and right side of the given expression.	It only evaluates the left sides of the given expression.
3	**Operates on**	It operates on Boolean data types as well as on bits.	It operates only on Boolean datatype.
4	**Uses**	Used to check logical condition and also used to mask off certain bits such as parity bits.	Used only to check the logic conditions.
5	**Example**	z = x & y	if (y > 1 && y > x)

Java Operators

- Post and pre increment

Operator	Operation
++ --	increment, decrement
+ -	unary plus, minus
!	boolean not
(<type>)	cast to <type>
* / %	multiplication, division, remainder
+ -	addition/concatenation, subtraction
< <= > >=	relational ordering
== !=	relational equality, inequality
&&	boolean and
||	boolean or
= += -= *= /= %=	assignments