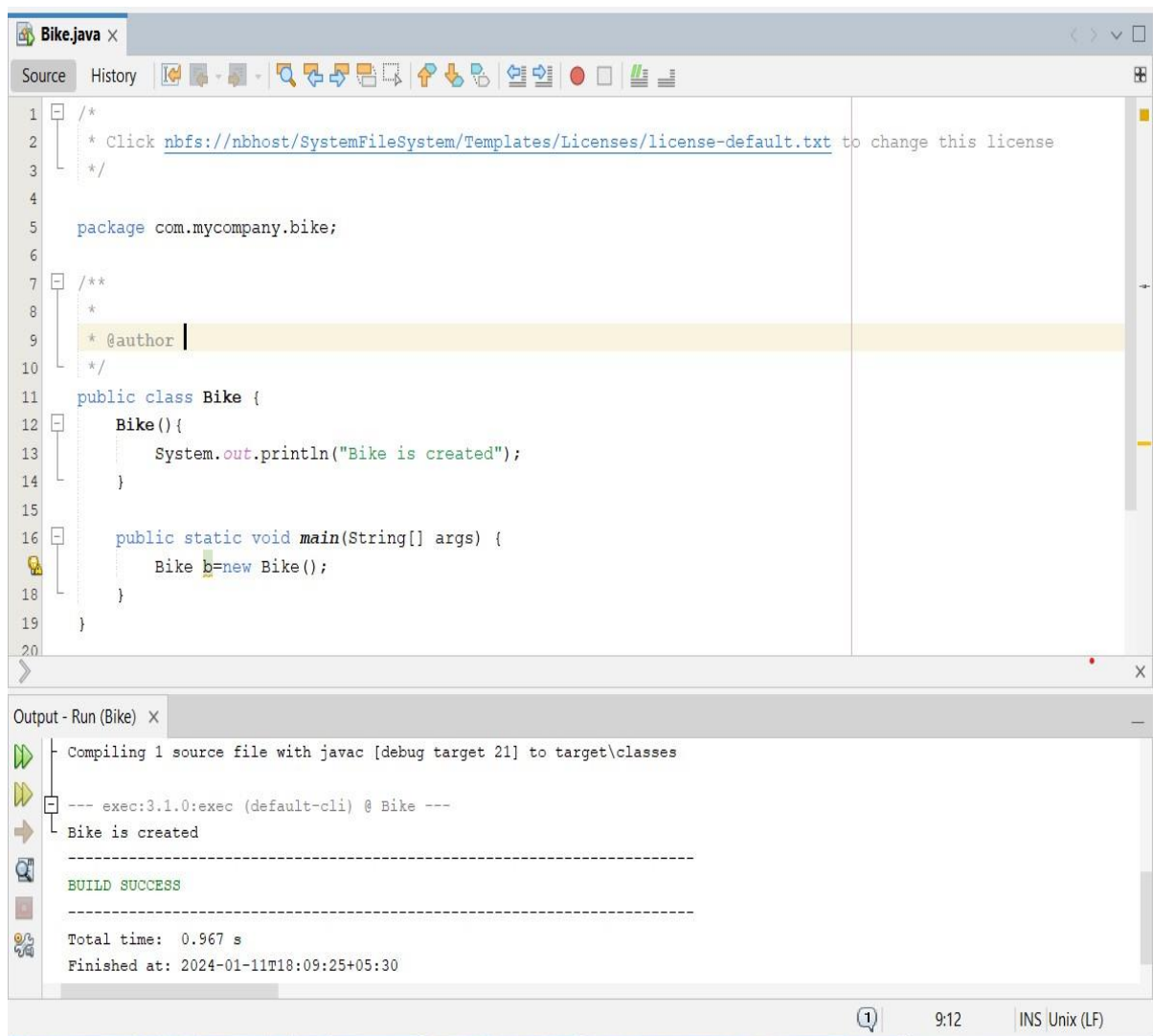


Practical 1: OOPs Concept in Java-1

- a. Write a program to create a class and implement default, overloaded and copy constructor.

Default Constructor



The screenshot displays an IDE window titled "Bike.java" with the following code:

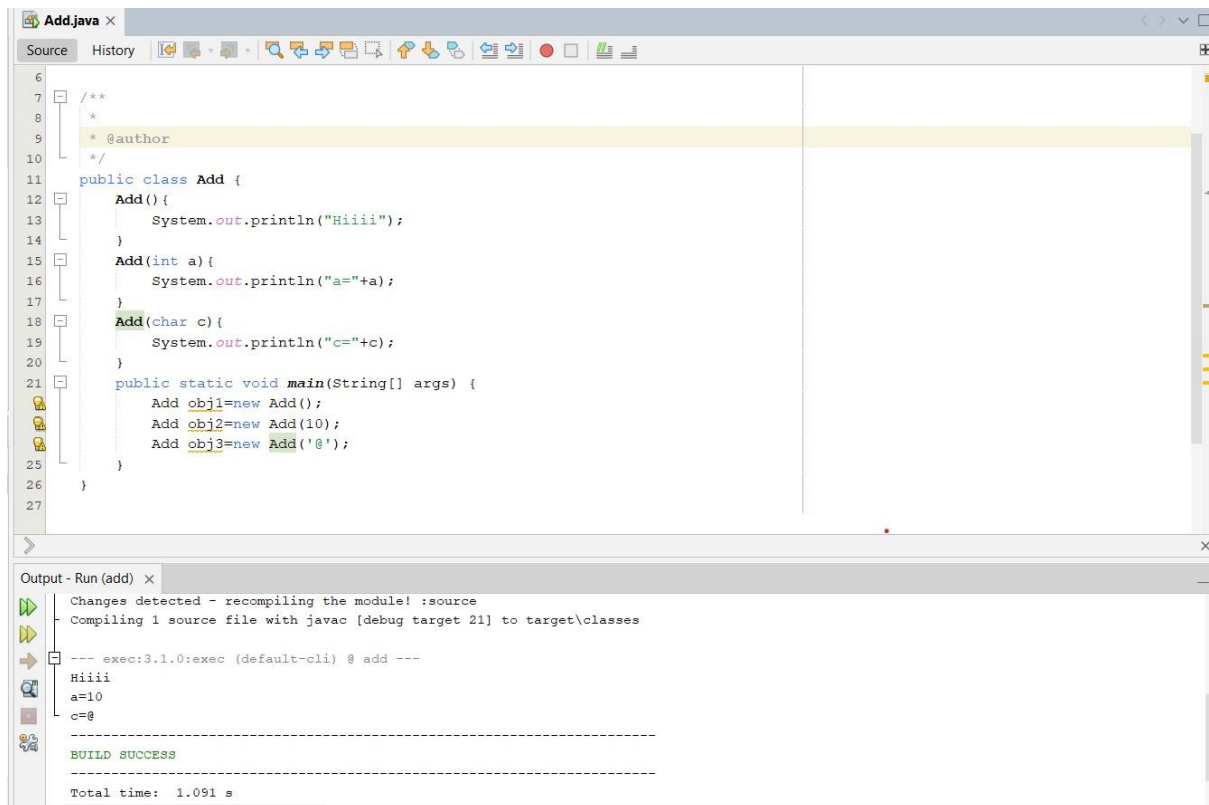
```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   */
4
5  package com.mycompany.bike;
6
7  /**
8   *
9   * @author
10  */
11 public class Bike {
12     Bike(){
13         System.out.println("Bike is created");
14     }
15
16     public static void main(String[] args) {
17         Bike b=new Bike();
18     }
19 }
20
```

Below the code editor, the "Output - Run (Bike)" window shows the execution results:

```
Compiling 1 source file with javac [debug target 21] to target\classes
--- exec:3.1.0:exec (default-cli) @ Bike ---
Bike is created
-----
BUILD SUCCESS
-----
Total time: 0.967 s
Finished at: 2024-01-11T18:09:25+05:30
```

The status bar at the bottom indicates the time is 9:12 and the environment is INS Unix (LF).

Overloaded Constructor



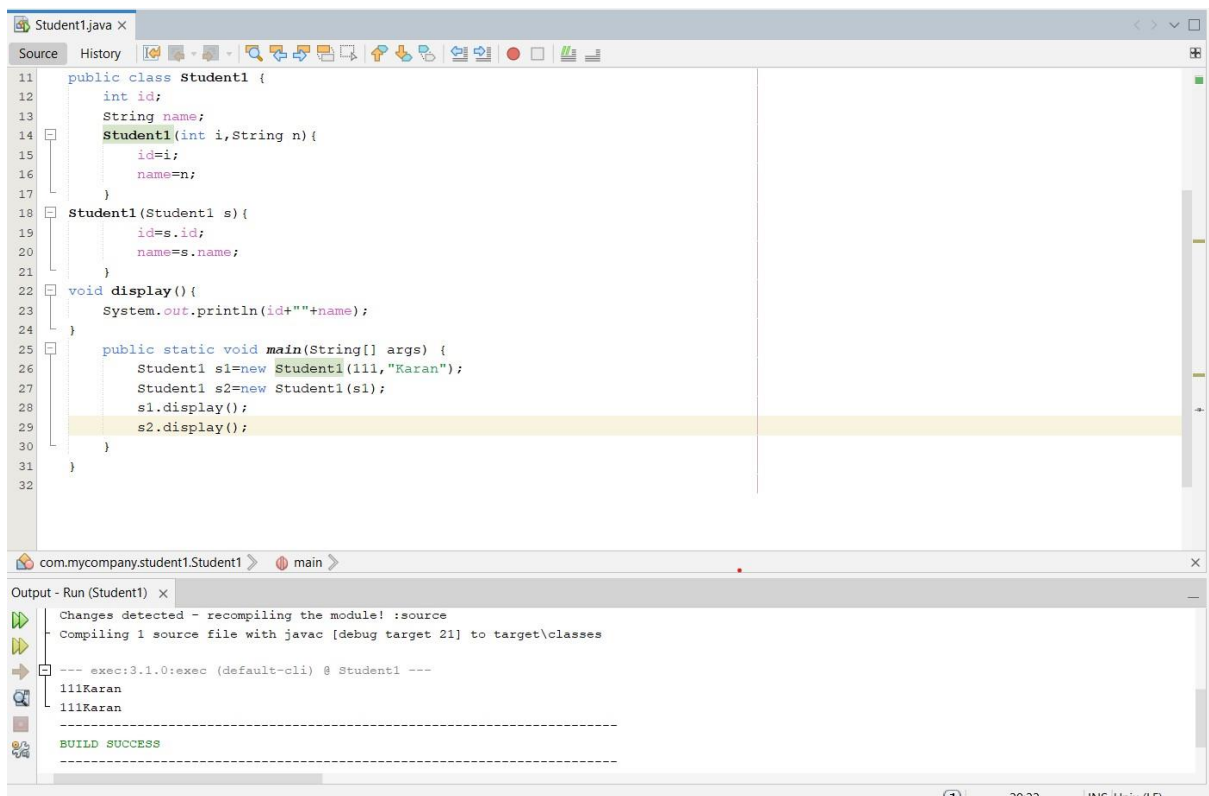
The screenshot shows an IDE with a file named `Add.java`. The code defines a class `Add` with three constructors: a default constructor, a constructor taking an `int`, and a constructor taking a `char`. The `main` method creates three objects: `obj1` (default), `obj2` (with `10`), and `obj3` (with `'@'`).

```
6
7 /**
8  *
9  * @author
10 */
11 public class Add {
12     Add() {
13         System.out.println("Hiiii");
14     }
15     Add(int a) {
16         System.out.println("a="+a);
17     }
18     Add(char c) {
19         System.out.println("c="+c);
20     }
21     public static void main(String[] args) {
22         Add obj1=new Add();
23         Add obj2=new Add(10);
24         Add obj3=new Add('@');
25     }
26 }
27
```

The output window shows the following execution results:

```
Output - Run (add) x
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 21] to target\classes
--- exec:3.1.0:exec (default-cli) @ add ---
Hiiii
a=10
c=@
BUILD SUCCESS
Total time: 1.091 s
```

Copy Constructor



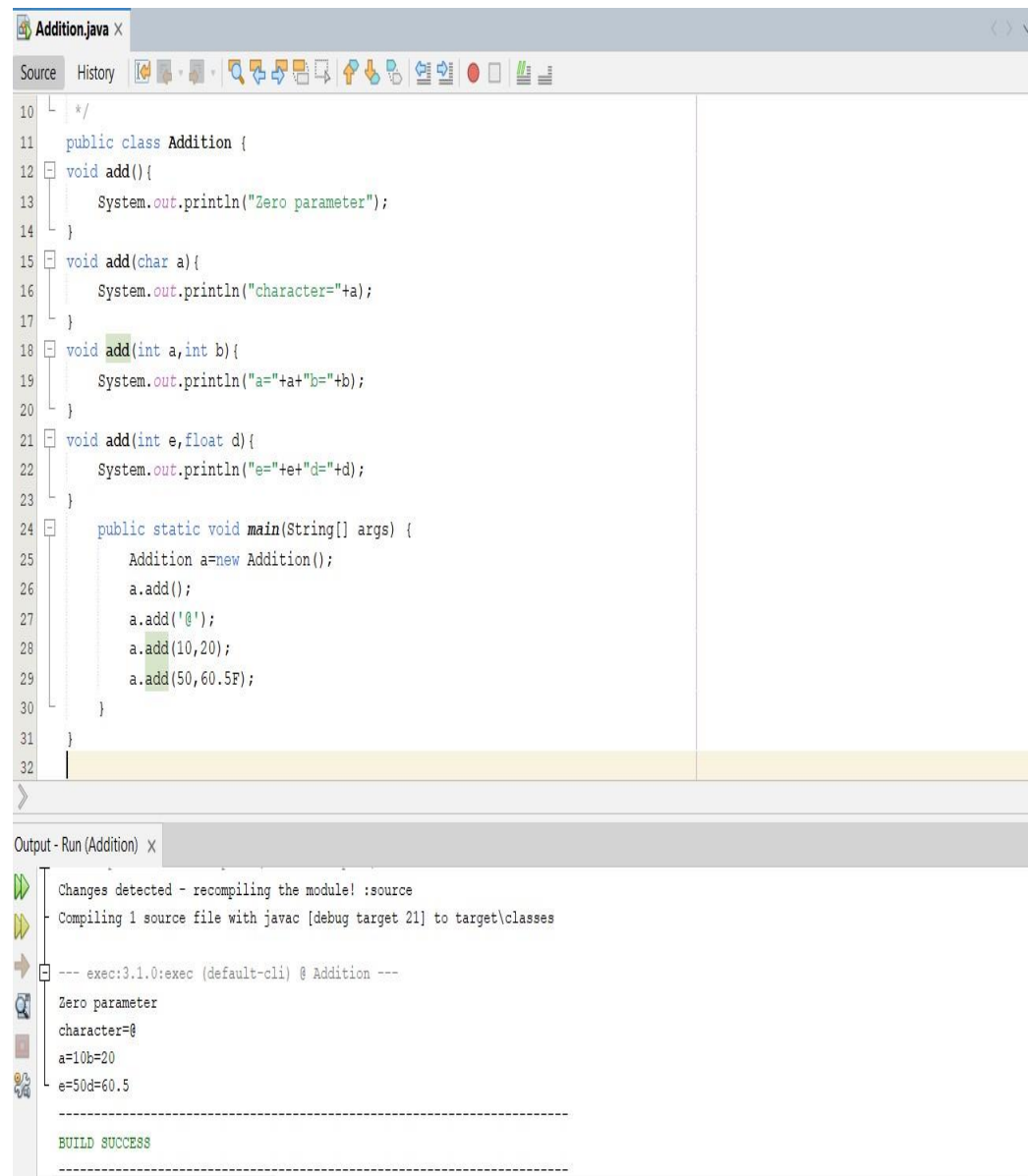
The screenshot shows an IDE with a file named `Student1.java`. The code defines a class `Student1` with a default constructor, a constructor taking `int` and `String`, and a copy constructor that takes a `Student1` object. The `main` method creates two objects: `s1` (with `111` and `"Karan"`) and `s2` (created by copying `s1`).

```
11 public class Student1 {
12     int id;
13     String name;
14     Student1(int i,String n){
15         id=i;
16         name=n;
17     }
18     Student1(Student1 s){
19         id=s.id;
20         name=s.name;
21     }
22     void display(){
23         System.out.println(id+" "+name);
24     }
25     public static void main(String[] args) {
26         Student1 s1=new Student1(111,"Karan");
27         Student1 s2=new Student1(s1);
28         s1.display();
29         s2.display();
30     }
31 }
32
```

The output window shows the following execution results:

```
Output - Run (Student1) x
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 21] to target\classes
--- exec:3.1.0:exec (default-cli) @ Student1 ---
111Karan
111Karan
BUILD SUCCESS
```

b. Write a program to create a class and implement the concept of Method Overloading.



The screenshot displays an IDE window titled "Addition.java" with a source code editor and an output console. The code defines a class named "Addition" with four overloaded methods: "add()" (no parameters), "add(char a)" (one char parameter), "add(int a, int b)" (two int parameters), and "add(int e, float d)" (one int and one float parameter). Each method prints a message indicating the parameters received. The "main" method creates an instance of the "Addition" class and calls each of the four "add" methods in sequence. The output console shows the execution results, confirming that each method is called with the correct number and type of arguments, demonstrating successful method overloading.

```
10  /*
11  public class Addition {
12  void add(){
13      System.out.println("Zero parameter");
14  }
15  void add(char a){
16      System.out.println("character="+a);
17  }
18  void add(int a,int b){
19      System.out.println("a="+a+"b="+b);
20  }
21  void add(int e,float d){
22      System.out.println("e="+e+"d="+d);
23  }
24  public static void main(String[] args) {
25      Addition a=new Addition();
26      a.add();
27      a.add('@');
28      a.add(10,20);
29      a.add(50,60.5F);
30  }
31  }
32
```

Output - Run (Addition) x

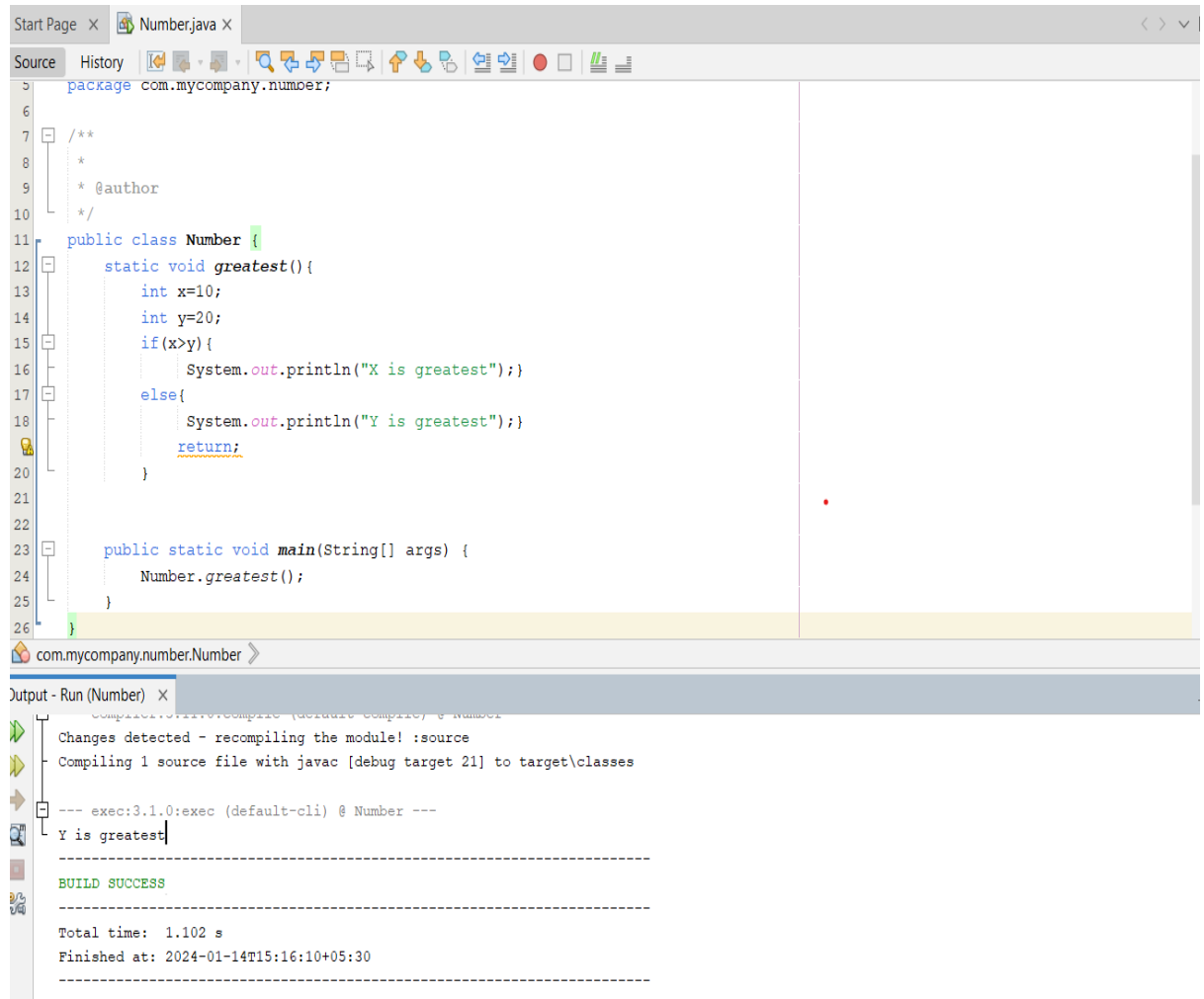
```
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ Addition ---

Zero parameter
character=@
a=10b=20
e=50d=60.5

-----
BUILD SUCCESS
-----
```

c. Write a program to create a class in implement the method of Static methods.



The screenshot shows an IDE window titled "Number.java" with the following code:

```
5 package com.mycompany.numoer;
6
7 /**
8  *
9  * @author
10 */
11 public class Number {
12     static void greatest(){
13         int x=10;
14         int y=20;
15         if(x>y){
16             System.out.println("X is greatest");
17         }
18         else{
19             System.out.println("Y is greatest");
20         }
21         return;
22     }
23
24     public static void main(String[] args) {
25         Number.greatest();
26     }
27 }
```

The output window shows the following text:

```
--- exec:3.1.0:exec (default-cli) @ Number ---
Y is greatest
BUILD SUCCESS
Total time: 1.102 s
Finished at: 2024-01-14T15:16:10+05:30
```