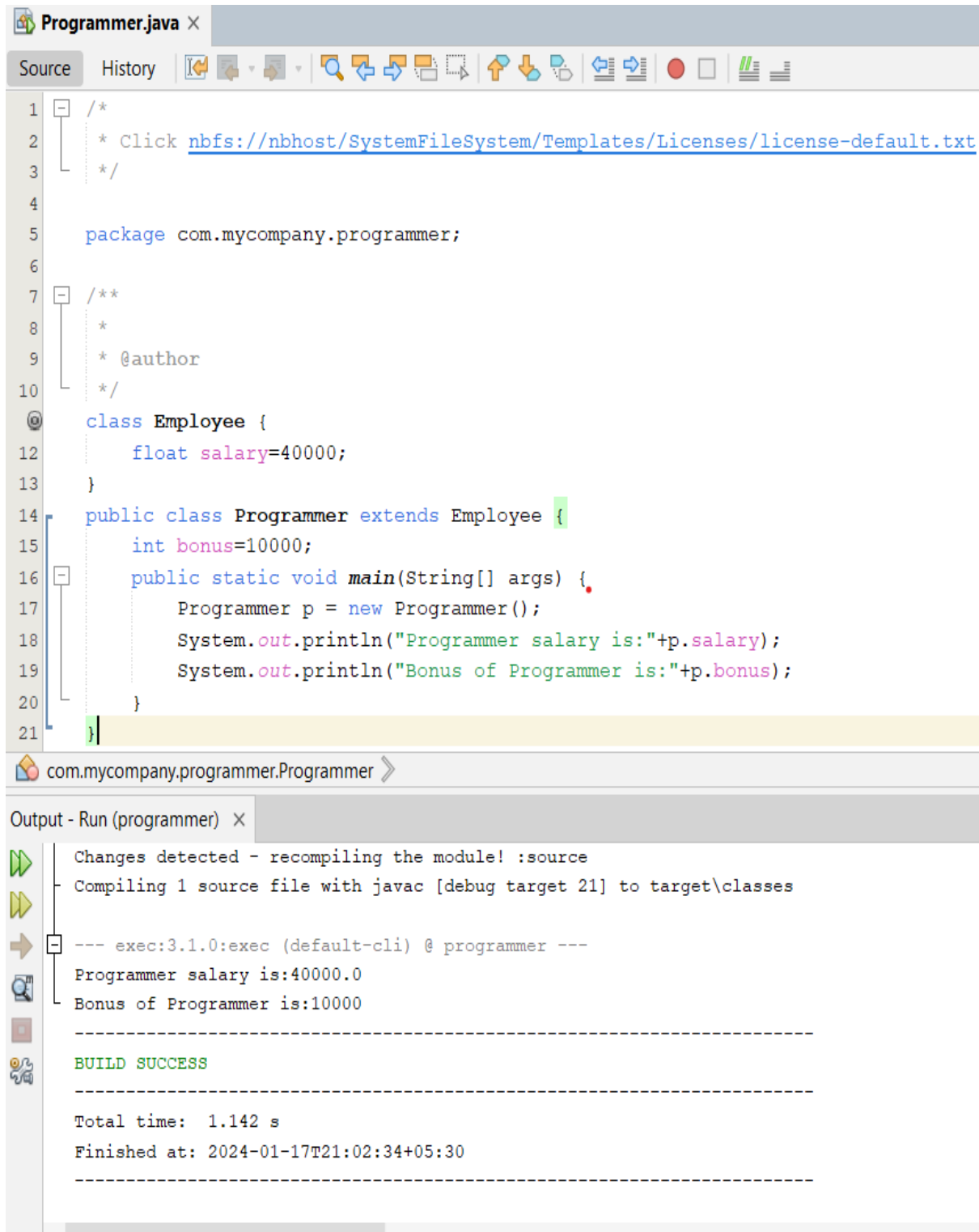


Practical 2 : OOPs concepts in Java-2

- a. Write a program to implement the concepts of Inheritance and Method Overriding.

Inheritance:



The screenshot displays an IDE with a Java file named `Programmer.java`. The code defines an `Employee` class with a `salary` attribute and a `Programmer` class that inherits from `Employee`, adding a `bonus` attribute. The `main` method in `Programmer` creates an instance and prints its attributes. Below the code editor, the 'Output - Run (programmer)' window shows the compilation and execution process, including the output of the `main` method and a 'BUILD SUCCESS' message.

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
3   */
4
5  package com.mycompany.programmer;
6
7  /**
8   *
9   * @author
10  */
11  @
12  class Employee {
13      float salary=40000;
14  }
15  public class Programmer extends Employee {
16      int bonus=10000;
17      public static void main(String[] args) {
18          Programmer p = new Programmer();
19          System.out.println("Programmer salary is:"+p.salary);
20          System.out.println("Bonus of Programmer is:"+p.bonus);
21      }
```

com.mycompany.programmer.Programmer >

Output - Run (programmer) 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) @ programmer ---
Programmer salary is:40000.0
Bonus of Programmer is:10000
-----
BUILD SUCCESS
-----
Total time: 1.142 s
Finished at: 2024-01-17T21:02:34+05:30
-----
```

Single Inheritance:

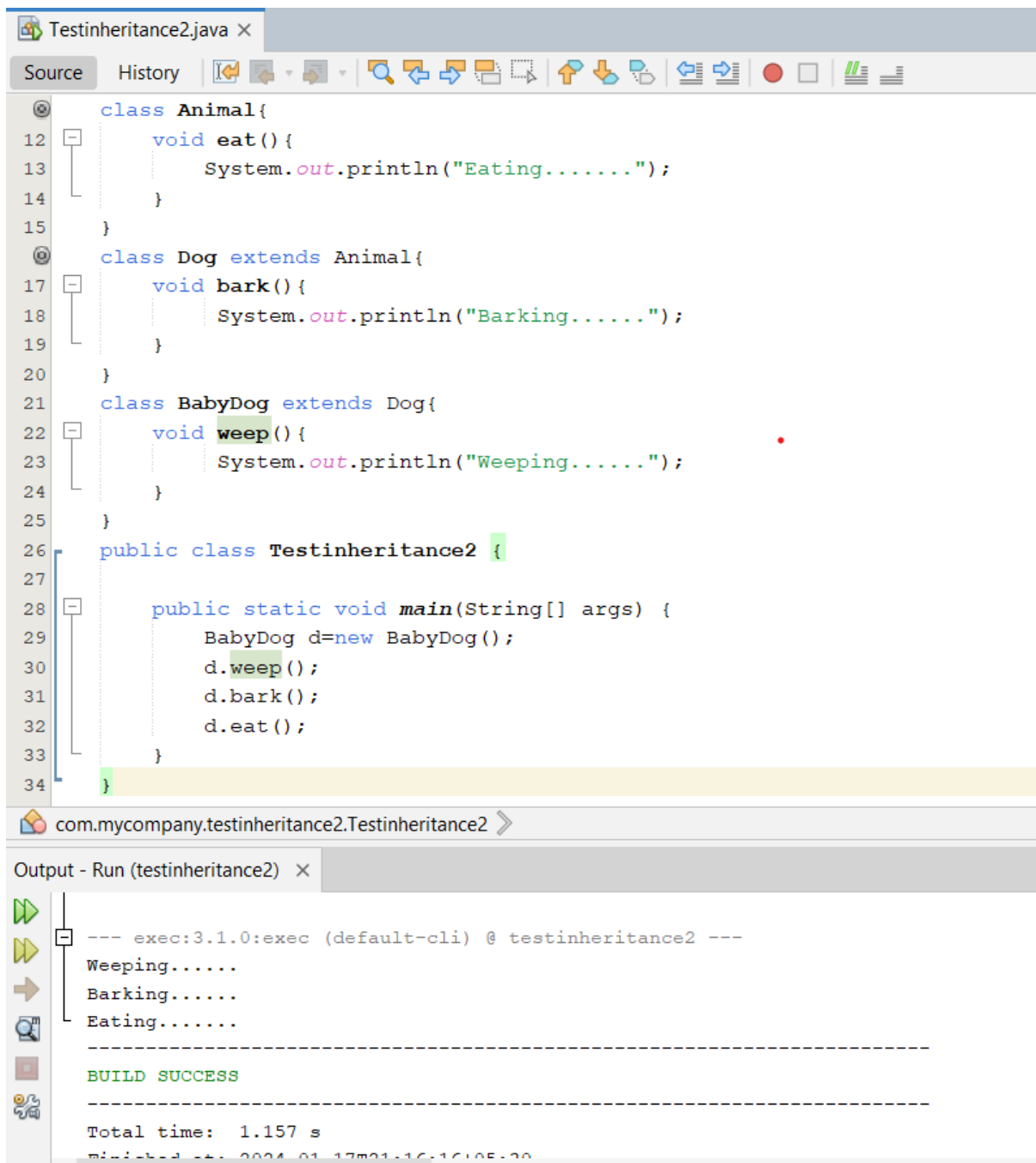
The screenshot displays an IDE window titled "Testinheritance.java" with a "Source" tab. The code defines an `Animal` class with an `eat()` method, a `Dog` class that inherits from `Animal` and adds a `bark()` method, and a `Testinheritance` class with a `main` method that creates a `Dog` object and calls both `bark()` and `eat()`.

```
7  /**
8   *
9   * @author
10  */
11  class Animal{
12      void eat(){
13          System.out.println("Eating.....");
14      }
15  }
16  class Dog extends Animal{
17      void bark(){
18          System.out.println("Barking.....");
19      }
20  }
21  public class Testinheritance {
22      public static void main(String[] args) {
23          Dog d=new Dog();
24          d.bark();
25          d.eat();
26      }
27  }
```

Below the code editor is the "Output - Run (testinheritance)" window. It shows the compilation process and the execution output, which matches the code's logic: "Barking....." followed by "Eating.....". The output also indicates a successful build and provides timing and completion information.

```
Compiling 1 source file with javac [debug target 21] to target\classes
--- exec:3.1.0:exec (default-cli) @ testinheritance ---
Barking.....
Eating.....
-----
BUILD SUCCESS
-----
Total time: 1.119 s
Finished at: 2024-01-17T21:11:04+05:30
-----
```

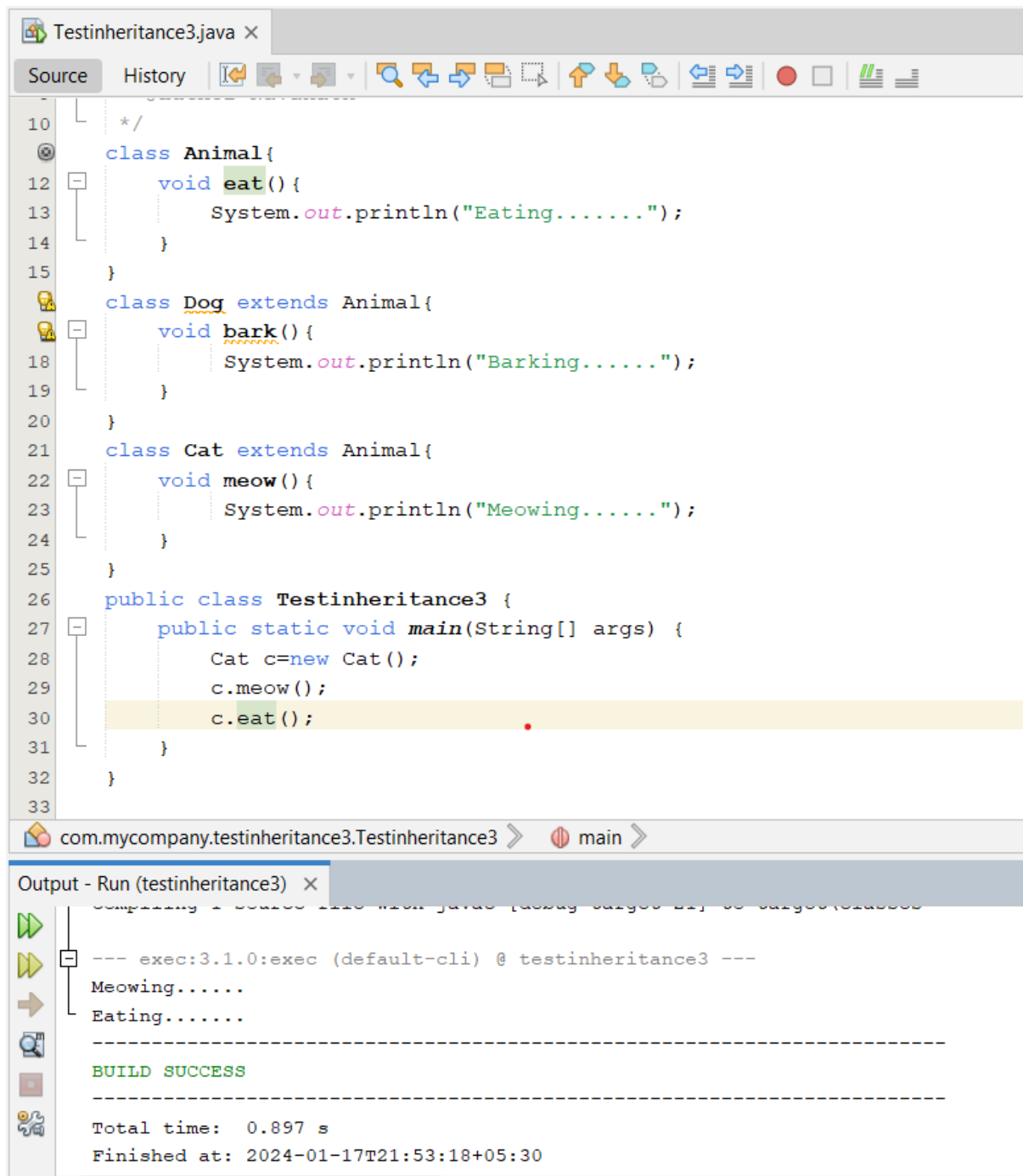
Multilevel Inheritance:



```
Testinheritance2.java x
Source History
class Animal{
12     void eat() {
13         System.out.println("Eating.....");
14     }
15 }
class Dog extends Animal{
17     void bark() {
18         System.out.println("Barking.....");
19     }
20 }
class BabyDog extends Dog{
22     void weep() {
23         System.out.println("Weeping.....");
24     }
25 }
public class Testinheritance2 {
26
27     public static void main(String[] args) {
28         BabyDog d=new BabyDog();
29         d.weep();
30         d.bark();
31         d.eat();
32     }
33 }
34

com.mycompany.testinheritance2.Testinheritance2 >
Output - Run (testinheritance2) x
--- exec:3.1.0:exec (default-cli) @ testinheritance2 ---
Weeping.....
Barking.....
Eating.....
-----
BUILD SUCCESS
-----
Total time: 1.157 s
Finished at: 2024-01-17T21:16:16+05:30
```

Hierarchical Inheritance:



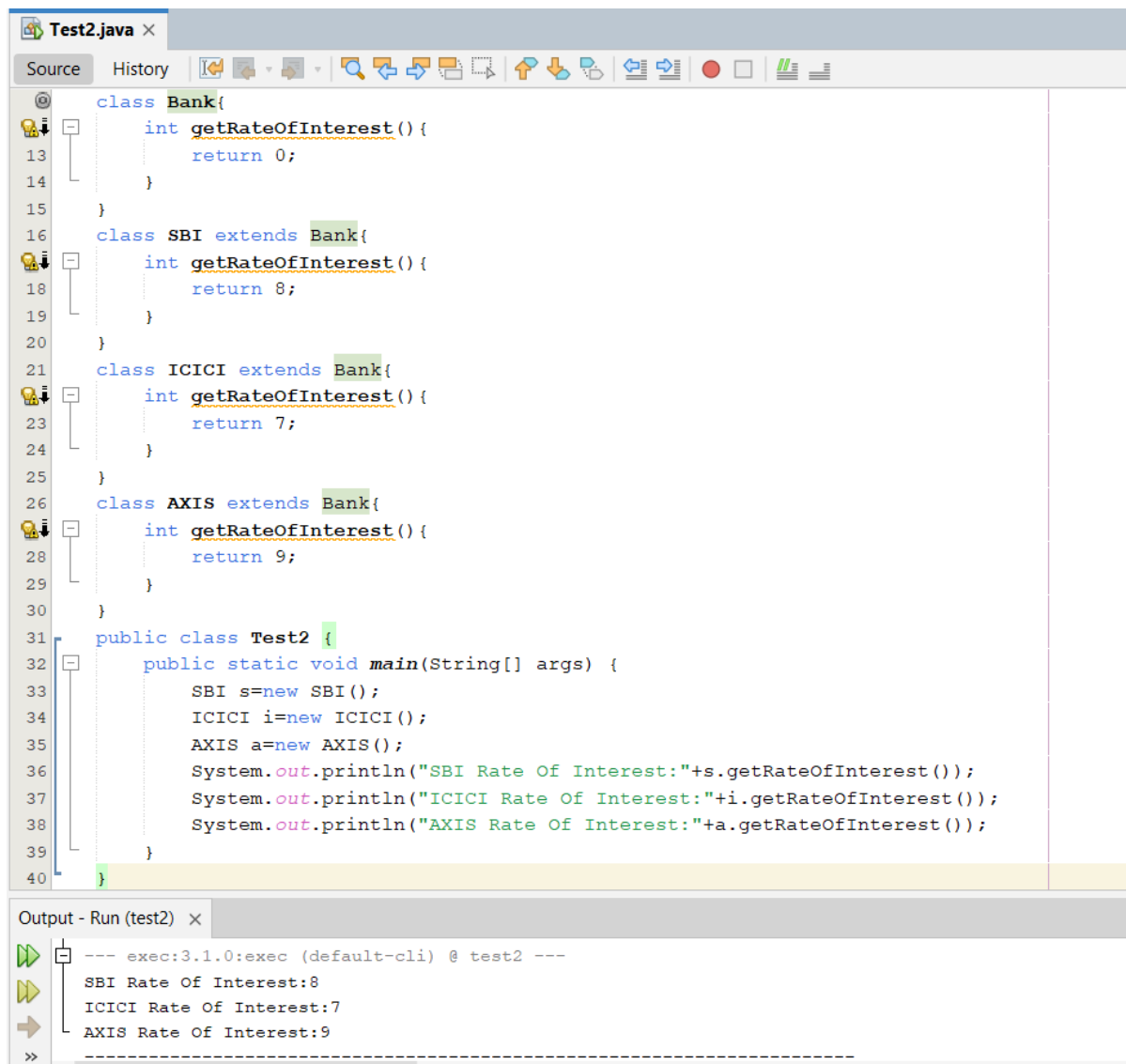
```
10  */
11  class Animal{
12      void eat(){
13          System.out.println("Eating.....");
14      }
15  }
16  class Dog extends Animal{
17      void bark(){
18          System.out.println("Barking.....");
19      }
20  }
21  class Cat extends Animal{
22      void meow(){
23          System.out.println("Meowing.....");
24      }
25  }
26  public class Testinheritance3 {
27      public static void main(String[] args) {
28          Cat c=new Cat();
29          c.meow();
30          c.eat();
31      }
32  }
33
```

com.mycompany.testinheritance3.Testinheritance3 > main >

Output - Run (testinheritance3) X

```
--- exec:3.1.0:exec (default-cli) @ testinheritance3 ---
Meowing.....
Eating.....
-----
BUILD SUCCESS
-----
Total time: 0.897 s
Finished at: 2024-01-17T21:53:18+05:30
```

Method Overriding:



The screenshot displays an IDE window titled "Test2.java" with a source code editor and an output console. The code defines a base class `Bank` with a `getRateOfInterest()` method returning 0. Three subclasses, `SBI`, `ICICI`, and `AXIS`, extend `Bank` and override the `getRateOfInterest()` method to return 8, 7, and 9 respectively. A `Test2` class contains a `main` method that creates instances of these classes and prints their interest rates. The output console shows the execution results: "SBI Rate Of Interest:8", "ICICI Rate Of Interest:7", and "AXIS Rate Of Interest:9".

```
class Bank{
    int getRateOfInterest(){
        return 0;
    }
}
class SBI extends Bank{
    int getRateOfInterest(){
        return 8;
    }
}
class ICICI extends Bank{
    int getRateOfInterest(){
        return 7;
    }
}
class AXIS extends Bank{
    int getRateOfInterest(){
        return 9;
    }
}
public class Test2 {
    public static void main(String[] args) {
        SBI s=new SBI();
        ICICI i=new ICICI();
        AXIS a=new AXIS();
        System.out.println("SBI Rate Of Interest:"+s.getRateOfInterest());
        System.out.println("ICICI Rate Of Interest:"+i.getRateOfInterest());
        System.out.println("AXIS Rate Of Interest:"+a.getRateOfInterest());
    }
}
```

Output - Run (test2) x

```
--- exec:3.1.0:exec (default-cli) @ test2 ---
SBI Rate Of Interest:8
ICICI Rate Of Interest:7
AXIS Rate Of Interest:9
>>
```

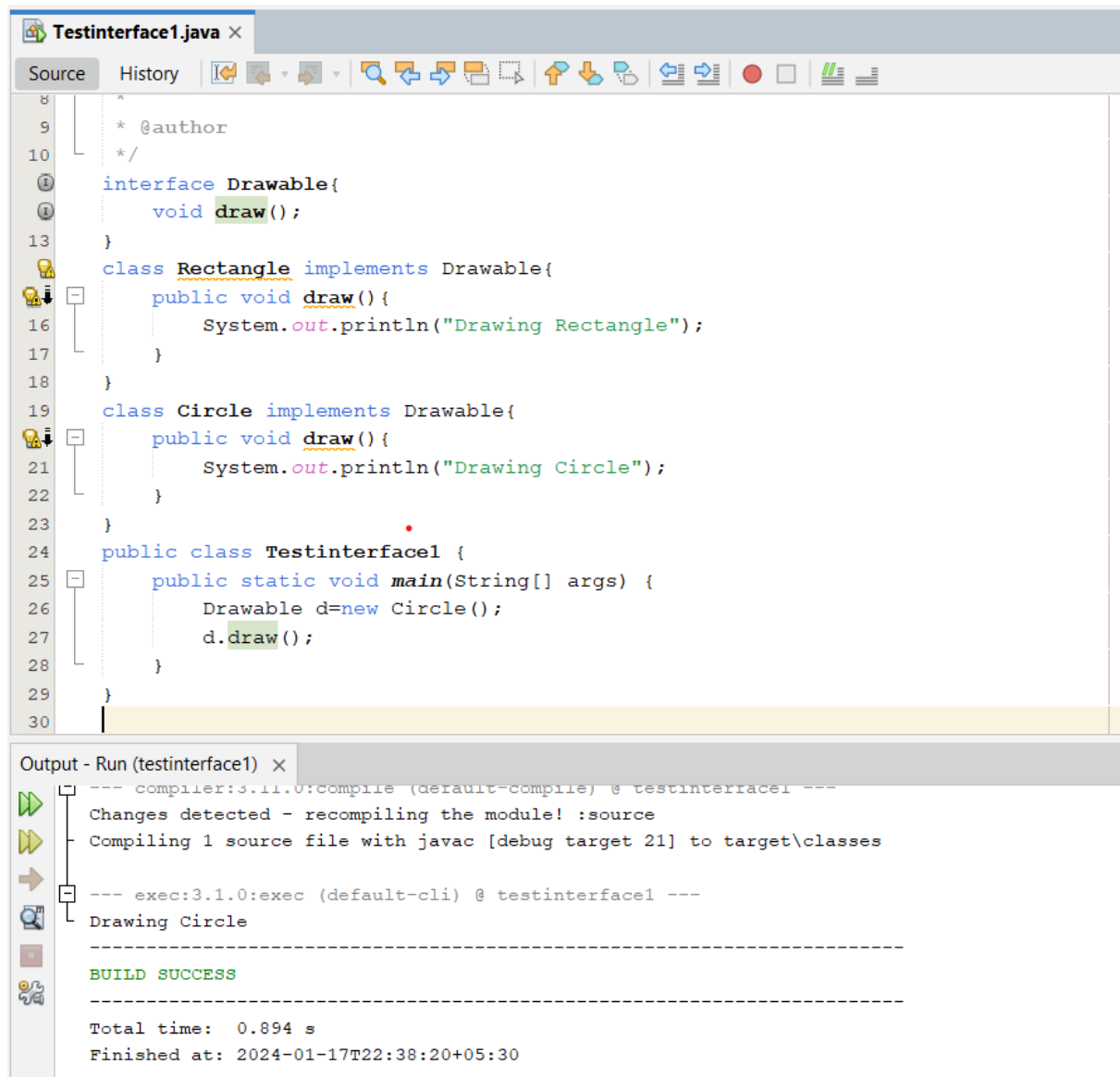
- b. Write a program to implement the concepts of Abstract classes and methods.

```
1  /**
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change t
3   */
4
5  package com.mycompany.honda4;
6
7  /**
8   *
9   * @author
10  */
11  abstract class Bike{
12      abstract void run();
13  }
14  public class Honda4 extends Bike {
15      void run(){
16          System.out.println("running safely...");
17      }
18      public static void main(String[] args) {
19          Bike obj=new Honda4();
20          obj.run();
21      }
22  }
23
```

Output - Run (honda4) x

```
--- compiler:3.11.0:compile (default-compile) @ honda4 ---
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) @ honda4 ---
running safely...
-----
BUILD SUCCESS
-----
Total time: 0.877 s
Finished at: 2024-01-17T22:23:56+05:30
-----
```

c. Write a program to implement the concept of Interfaces.



The screenshot displays an IDE window titled "Testinterface1.java". The code defines an interface `Drawable` with a `draw()` method. Two classes, `Rectangle` and `Circle`, implement this interface. The `Circle` class's `draw()` method prints "Drawing Circle". A `Testinterface1` class contains a `main` method that creates a `Circle` object and calls its `draw()` method. The output window shows the compilation and execution process, resulting in "BUILD SUCCESS" and the printed output "Drawing Circle".

```
8      * @author
9      */
10     interface Drawable{
11         void draw();
12     }
13
14     class Rectangle implements Drawable{
15         public void draw(){
16             System.out.println("Drawing Rectangle");
17         }
18     }
19
20     class Circle implements Drawable{
21         public void draw(){
22             System.out.println("Drawing Circle");
23         }
24     }
25
26     public class Testinterface1 {
27         public static void main(String[] args) {
28             Drawable d=new Circle();
29             d.draw();
30         }
31     }
```

Output - Run (testinterface1) x

```
--- compiler:3.11.0:compile (default-compile) @ testinterface1 ---
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) @ testinterface1 ---
Drawing Circle

-----
BUILD SUCCESS
-----

Total time: 0.894 s
Finished at: 2024-01-17T22:38:20+05:30
-----
```