



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGIES

FALL 2023

CA EXAMINATION

NAME: NDE HURICH DILAN

MATRICULE: ICTU20223351

EMAIL: nde.dilan@ictuniversity.edu.cm

TEL: +237694525931

JAVA CA

PART1

Java Interface

-
- 1) Write a Java program to create an interface Shape with the getArea() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getArea() method for each of the three classes.

```

package main.java.com.example;

interface Shape {
    double getArea();
}

class Rectangle implements Shape {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    @Override
    public double getArea() {
        return width * height;
    }
}

class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }
}

class Triangle implements Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
    public double getArea() {
        return 0.5 * base * height;
    }
}

```

Output

```

run:
Rectangle Area: 15.0
Circle Area: 50.3
Triangle Area: 6.0
BUILD SUCCESSFUL (total time: 1 second)

```

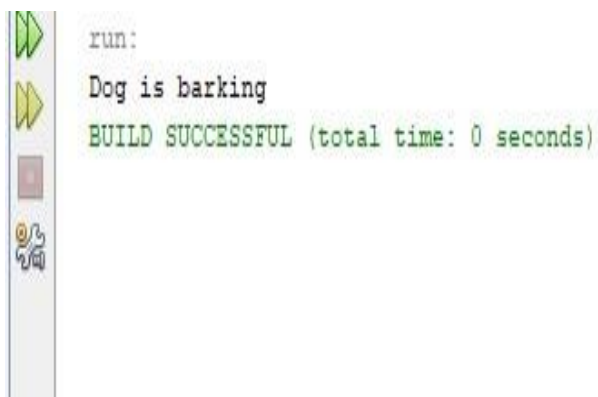
- 2) Write a Java program to create a `Animal` interface with a method called `bark()` that takes no arguments and returns `void`. Create a `Dog` class that implements `Animal` and overrides `speak()` to print "Dog is barking".

```
package main.java.com.example;    Test.java is a non-project
interface Animal {
    void bark();
}

class Dog implements Animal {

    @Override
    public void bark() {
        System.out.println(x:"Dog is barking");
    }
}
```

Output



```
run:
Dog is barking
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 3) Write a Java program to create an interface `Flyable` with a method called `fly_obj()`. Create three classes `Spacecraft`, `Airplane`, and `Helicopter` that implement the `Flyable` interface. Implement the `fly_obj()` method for each of the three classes.

```

1 package main.java.com.example; Test.java is a non-project file, only syntax errors are reported
2
3
4 interface Flyable {
5     void fly_obj();
6 }
7
8 class Spacecraft implements Flyable {
9
10     @Override
11     public void fly_obj() {
12         System.out.println(x:"Spacecraft is flying");
13     }
14 }
15
16 class Airplane implements Flyable {
17
18     @Override
19     public void fly_obj() {
20         System.out.println(x:"Airplane is flying");
21     }
22 }
23
24 class Helicopter implements Flyable {
25
26     @Override
27     public void fly_obj() {
28         System.out.println(x:"Helicopter is flying");
29     }
30 }





```

```

26     public void fly_obj() {
27         System.out.println("Helicopter is flying in the air");
28     }
29
30 // Main class
31 public class Main {
32     public static void main(String[] args) {
33         Spacecraft spaceship = new Spacecraft();
34         Airplane airplane = new Airplane();
35         Helicopter helicopter = new Helicopter();
36
37         spaceship.fly_obj();
38         airplane.fly_obj();
39         helicopter.fly_obj();
40     }
41 }

```

Output

```
run:
Spacecraft is flying in space
Airplane is flying in the sky
Helicopter is flying in the air
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 4) Write a Java programming to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw, calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods.

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  // Account Interface
5  interface Account {
6      void deposit(double amount);
7      void withdraw(double amount);
8      double calculateInterest();
9      void viewBalance();
10 }
11
12 // Class Bank
13 class Bank {
14     private List<Account> accounts;
15
16     public Bank() {
17         this.accounts = new ArrayList<>();
18     }
19 }
```

```

19
20     public void addAccount(Account account) {
21         accounts.add(account);
22     }
23 }
24
25 // Class SavingsAccount
26 class SavingsAccount implements Account {
27     private double balance;
28     private double interestRate;
29
30     public SavingsAccount(double initialBalance, double interestRate) {
31         this.balance = initialBalance;
32         this.interestRate = interestRate;
33     }
34
35     @Override
36     public void deposit(double amount) {
37         balance += amount;
38     }
39
40     @Override
41     public void withdraw(double amount) {
42         if (amount <= balance) {
43             balance -= amount;
44         } else {
45             System.out.println("Insufficient funds");
46         }
47     }
48
49     @Override
50     public double calculateInterest() {
51         return balance * interestRate;
52     }
53
54     @Override
55     public void viewBalance() {

```

```

56         System.out.println("Savings Account Balance: " + balance);
57     }
58
59     public void addInterest() {
60         balance += calculateInterest();
61     }
62 }
63
64 // Class CurrentAccount
65 class CurrentAccount implements Account {
66     private double balance;
67
68     public CurrentAccount(double initialBalance) {
69         this.balance = initialBalance;
70     }
71
72     @Override
73     public void deposit(double amount) {

```

```

74     public void deposit(double amount) {
75         balance += amount;
76     }
77
78     @Override
79     public void withdraw(double amount) {
80         if (amount <= balance) {
81             balance -= amount;
82         } else {
83             System.out.println("Insufficient funds");
84         }
85     }
86
87     @Override
88     public double calculateInterest() {
89         return 0;
90     }
91
92     @Override
93     public void viewBalance() {
94         System.out.println("Current Account Balance: " + balance);
95     }
96
97     public void overdraftCharge() {
98         balance -= 10; // Example overdraft charge
99     }
100
101 // Main class
102 public class Main {
103     public static void main(String[] args) {
104         Bank bank = new Bank();
105
106         SavingsAccount savingsAccount = new SavingsAccount(1000, 0.05);
107         CurrentAccount currentAccount = new CurrentAccount(500);
108
109         bank.addAccount(savingsAccount);

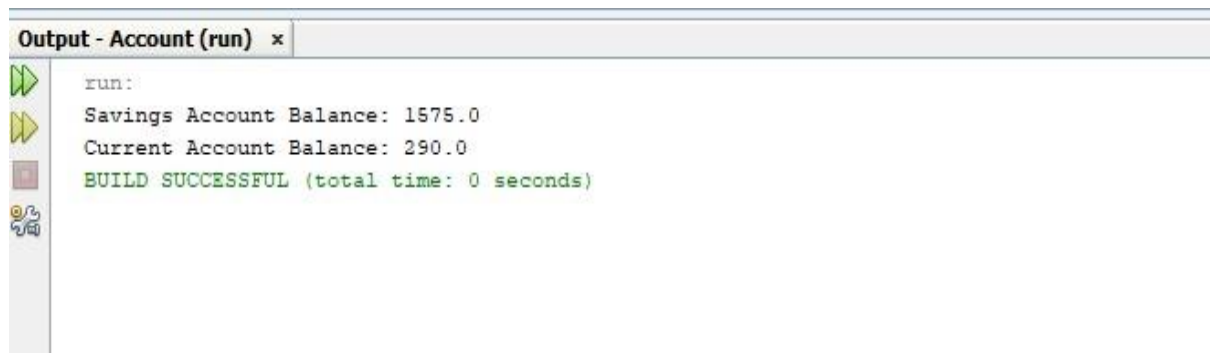
```

```

102 public class Main {
103     public static void main(String[] args) {
104         Bank bank = new Bank();
105
106         SavingsAccount savingsAccount = new SavingsAccount(1000, 0.05);
107         CurrentAccount currentAccount = new CurrentAccount(500);
108
109         bank.addAccount(savingsAccount);
110         bank.addAccount(currentAccount);
111
112         savingsAccount.deposit(500);
113         savingsAccount.addInterest();
114         savingsAccount.viewBalance();
115
116         currentAccount.withdraw(200);
117         currentAccount.overdraftCharge();
118         currentAccount.viewBalance();
119     }
120 }

```


Output



```
Output - Account (run) x
run:
Savings Account Balance: 1575.0
Current Account Balance: 290.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 5) Write a Java program to create an interface Resizable with methods `resizeWidth(int width)` and `resizeHeight(int height)` that allow an object to be resized. Create a class `Rectangle` that implements the `Resizable` interface and implements the resize methods.

```
1 // Resizable interface
2 interface Resizable {
3     void resizeWidth(int width);
4     void resizeHeight(int height);
5 }
6
7 // Rectangle class implementing Resizable interface
8 class Rectangle implements Resizable {
9     private int width;
10    private int height;
11
12    public Rectangle(int width, int height) {
13        this.width = width;
14        this.height = height;
15    }
16
17    @Override
18    public void resizeWidth(int width) {
19        this.width = width;
20    }
21}
```









```

20     }
21
22     @Override
23     public void resizeHeight(int height) {
24         this.height = height;
25     }
26
27     public void display() {
28         System.out.println("Rectangle Width: " + width);
29         System.out.println("Rectangle Height: " + height);
30     }
31 }
32
33 public class Main {
34     public static void main(String[] args) {
35         Rectangle rectangle = new Rectangle(5, 10);
36         System.out.println("Before resizing:");
37         rectangle.display();
38
39         rectangle.resizeWidth(8);
40         rectangle.resizeHeight(12);
41
42         System.out.println("After resizing:");
43         rectangle.display();
44     }
45 }

```

Output

Output - Resizable (run) x	
	run:
	Before resizing:
	Rectangle Width: 5
	Rectangle Height: 10
	After resizing:
	Rectangle Width: 8
	Rectangle Height: 12
	BUILD SUCCESSFUL (total time: 0 seconds)

- 6) Write a Java program to create an interface Drawable with a method draw() that takes no arguments and returns void. Create three classes Circle, Rectangle, and Triangle that implement the Drawable interface and override the draw() method to draw their respective shapes.






```
1 // Drawable interface
2 interface Drawable {
3     void draw();
4 }
5
6 // Circle class implementing Drawable interface
7 class Circle implements Drawable {
8     @Override
9     public void draw() {
10         System.out.println("Drawing a Circle");
11     }
12 }
13
14 // Rectangle class implementing Drawable interface
15 class Rectangle implements Drawable {
16     @Override
17     public void draw() {
18         System.out.println("Drawing a Rectangle");
19     }
20 }
21
22 // Triangle class implementing Drawable interface
23 class Triangle implements Drawable {
24     @Override
25     public void draw() {
26         System.out.println("Drawing a Triangle");
27     }
28 }
29
30 public class Main {
31     public static void main(String[] args) {
32         Drawable circle = new Circle();
33         Drawable rectangle = new Rectangle();
34         Drawable triangle = new Triangle();
```

```

29
30 public class Main {
31     public static void main(String[] args) {
32         Drawable circle = new Circle();
33         Drawable rectangle = new Rectangle();
34         Drawable triangle = new Triangle();
35
36         System.out.println("Drawing shapes:");
37         circle.draw();
38         rectangle.draw();
39         triangle.draw();
40     }
41 }

```

Output

Output - Drawable (run) x	
	run:
	Drawing shapes:
	Drawing a Circle
	Drawing a Rectangle
	Drawing a Triangle
BUILD SUCCESSFUL (total time: 0 seconds)	

- 7) Write a Java program to create an interface Sortable with a method sort() that sorts an array of integers in ascending order. Create two classes BubbleSort and SelectionSort that implement the Sortable interface and provide their own implementations of the sort() method.

```

1 // Sortable interface
2 interface Sortable {
3     void sort(int[] arr);
4 }
5
6 // BubbleSort class implementing Sortable interface
7 class BubbleSort implements Sortable {
8     @Override
9     public void sort(int[] arr) {
10         int n = arr.length;
11         for (int i = 0; i < n-1; i++) {
12             for (int j = 0; j < n-i-1; j++) {
13                 if (arr[j] > arr[j+1]) {
14                     // Swap elements
15                     int temp = arr[j];
16                     arr[j] = arr[j+1];
17                     arr[j+1] = temp;
18                 }
19             }
20         }
21     }
22 }

```

```

19 }
20 }
21 }
22 }
23
24 // SelectionSort class implementing Sortable interface
25 class SelectionSort implements Sortable {
26     @Override
27     public void sort(int[] arr) {
28         int n = arr.length;
29         for (int i = 0; i < n-1; i++) {
30             int minIndex = i;
31             for (int j = i+1; j < n; j++) {
32                 if (arr[j] < arr[minIndex]) {
33                     minIndex = j;
34                 }
35             }
36             // Swap elements
37             int temp = arr[minIndex];

```

```

38         arr[minIndex] = arr[i];
39         arr[i] = temp;
40     }
41 }
42
43
44 public class Main {
45     public static void main(String[] args) {
46         int[] arr = {29, 10, 14, 37, 13};
47
48         Sortable bubbleSort = new BubbleSort();
49         Sortable selectionSort = new SelectionSort();
50
51         System.out.println("Before sorting:");
52         printArray(arr);
53
54         bubbleSort.sort(arr);
55         System.out.println("After Bubble Sort:");

```

```

52         printArray(arr);
53
54         bubbleSort.sort(arr);
55         System.out.println("After Bubble Sort:");
56         printArray(arr);
57
58         selectionSort.sort(arr);
59         System.out.println("After Selection Sort:");
60         printArray(arr);
61     }
62
63     public static void printArray(int[] arr) {
64         for (int num : arr) {
65             System.out.print(num + " ");
66         }
67         System.out.println();
68     }
69 }

```

Output

```

Output - Sortable (run) x
run:
Before sorting:
29 10 14 37 13
After Bubble Sort:
10 13 14 29 37
After Selection Sort:
10 13 14 29 37
BUILD SUCCESSFUL (total time: 0 seconds)

```

- 8) Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball,

and Basketball that implement the Playable interface and override the play() method to play the respective sports.

```
1 // Playable interface
2 interface Playable {
3     void play();
4 }
5
6 // Football class implementing Playable interface
7 class Football implements Playable {
8     @Override
9     public void play() {
10         System.out.println("Playing Football");
11     }
12 }
13
14 // Volleyball class implementing Playable interface
15 class Volleyball implements Playable {
16     @Override
17     public void play() {
18         System.out.println("Playing Volleyball");
19     }
20 }
```

```
22 // Basketball class implementing Playable interface
23 class Basketball implements Playable {
24     @Override
25     public void play() {
26         System.out.println("Playing Basketball");
27     }
28 }
29
30 public class Main {
31     public static void main(String[] args) {
32         Playable football = new Football();
33         Playable volleyball = new Volleyball();
34         Playable basketball = new Basketball();
35
36         football.play();
37         volleyball.play();
38         basketball.play();
39     }
40 }
```

Output

```
Output - Playable (run) x
run:
Playing Football
Playing Volleyball
Playing Basketball
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 9) Write a Java program to create an interface Searchable with a method search(String keyword) that searches for a given keyword in a text document. Create two classes Document and WebPage that implement the Searchable interface and provide their own implementations of the search() method.

```
1 // Define the Searchable interface
2 interface Searchable {
3     void search(String keyword);
4 }
5
6 // Implementing class for Document
7 class Document implements Searchable {
8     private String content;
9
10    public Document(String content) {
11        this.content = content;
12    }
13
14    @Override
15    public void search(String keyword) {
16        if (content.contains(keyword)) {
17            System.out.println("Keyword " + keyword + " found in the document.");
18        } else {
```



```

19         System.out.println("Keyword '" + keyword + "' not found in the document.");
20     }
21 }
22 }
23
24 // Implementing class for WebPage
25 class WebPage implements Searchable {
26     private String url;
27     private String htmlContent;
28
29     public WebPage(String url, String htmlContent) {
30         this.url = url;
31         this.htmlContent = htmlContent;
32     }
33
34     @Override
35     public void search(String keyword) {
36         if (htmlContent.contains(keyword)) {

```

```

37             System.out.println("Keyword '" + keyword + "' found in the web page " + url + ".");
38         } else {
39             System.out.println("Keyword '" + keyword + "' not found in the web page " + url + ".");
40         }
41     }
42 }
43
44 public class Main {
45     public static void main(String[] args) {
46         // Create a Document and a WebPage
47         Document document = new Document("This is a sample document for testing.");
48         WebPage webPage = new WebPage("http://example.com", "<html><body>This is a sample web page for testing.</>");
49
50         // Search for keywords in Document and WebPage
51         document.search("sample");
52         webPage.search("web");
53     }
54 }

```

Output

```

Output - Searchable (run) x
run:
Keyword 'sample' found in the document.
Keyword 'web' found in the web page http://example.com.
BUILD SUCCESSFUL (total time: 0 seconds)

```

- 10) Write a Java program to create an interface `Encryptable` with methods `encrypt (String data)` and `decrypt (String encryptedData)` that define encryption and decryption operations. Create two classes `AES` and `RSA` that implement the

Encryptable interface and provide their own encryption and decryption algorithms.

```
1 // Encryptable interface
2 interface Encryptable {
3     String encrypt(String data);
4     String decrypt(String encryptedData);
5 }
6
7 // AES class implementing Encryptable interface
8 class AES implements Encryptable {
9     @Override
10    public String encrypt(String data) {
11        // Encryption using AES algorithm
12        return "AES encrypted data: " + data;
13    }
14
15    @Override
16    public String decrypt(String encryptedData) {
17        // Decryption using AES algorithm
18        return "AES decrypted data: " + encryptedData;
19    }
20 }
```





```
21
22 // RSA class implementing Encryptable interface
23 class RSA implements Encryptable {
24     @Override
25    public String encrypt(String data) {
26        // Encryption using RSA algorithm
27        return "RSA encrypted data: " + data;
28    }
29
30    @Override
31    public String decrypt(String encryptedData) {
32        // Decryption using RSA algorithm
33        return "RSA decrypted data: " + encryptedData;
34    }
35 }
36
37 public class Main {
```

```

38 public static void main(String[] args) {
39     Encryptable aes = new AES();
40     Encryptable rsa = new RSA();
41
42     String data = "This is a secret message";
43
44     String aesEncryptedData = aes.encrypt(data);
45     System.out.println(aesEncryptedData);
46
47     String aesDecryptedData = aes.decrypt(aesEncryptedData);
48     System.out.println(aesDecryptedData);
49
50     String rsaEncryptedData = rsa.encrypt(data);
51     System.out.println(rsaEncryptedData);
52
53     String rsaDecryptedData = rsa.decrypt(rsaEncryptedData);
54     System.out.println(rsaDecryptedData);
55 }
56 }

```

Output

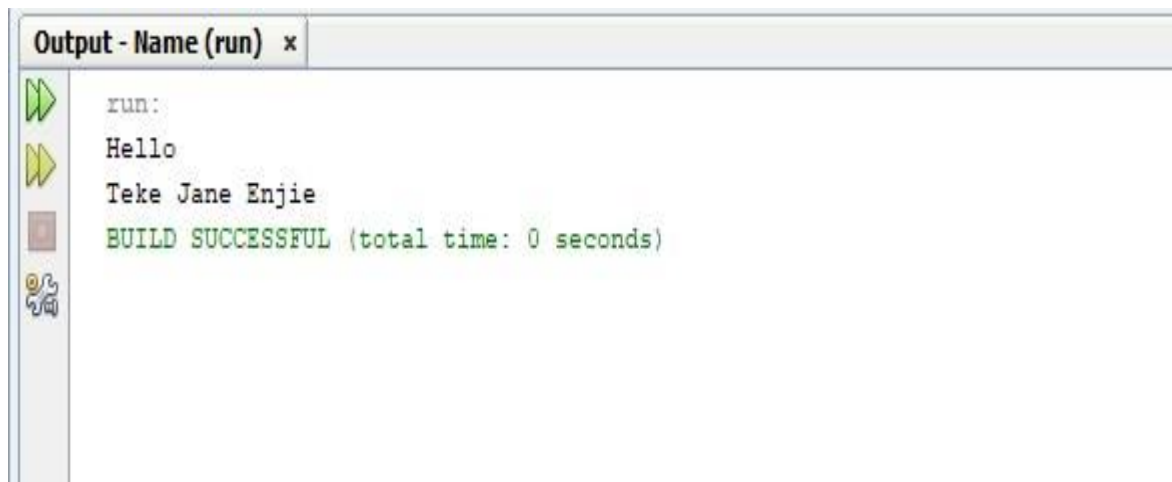
Output - Encryptable (run) ×	
	run:
	AES encrypted data: This is a secret message
	AES decrypted data: AES encrypted data: This is a secret message
	RSA encrypted data: This is a secret message
	RSA decrypted data: RSA encrypted data: This is a secret message
	BUILD SUCCESSFUL (total time: 0 seconds)

PART2

1. Write a Java program to print 'Hello' on screen and your name on a separate line.
Expected Output : Hello Alexandra Abramov

```
1 public class HelloName {
2     public static void main(String[] args) {
3         System.out.println("Hello");
4         System.out.println("Teke Jane Enjie");
5     }
6 }
7
```

Output

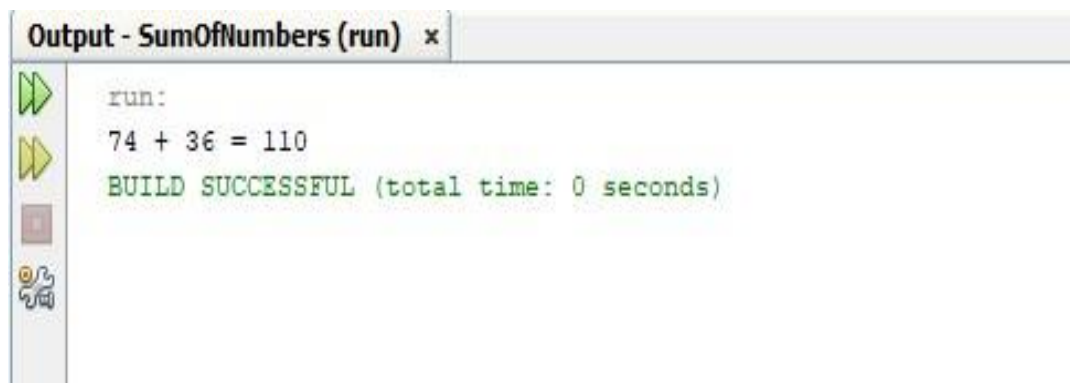


```
Output - Name (run) x
run:
Hello
Teke Jane Enjie
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Write a Java program to print the sum of two numbers. Test Data: 74 + 36 Expected Output : 110

```
1 public class SumOfNumbers {
2     public static void main(String[] args) {
3         // Define the two numbers to be added
4         int num1 = 74;
5         int num2 = 36;
6
7         // Calculate the sum of the two numbers
8         int sum = num1 + num2;
9
10        // Print the equation and the result
11        System.out.println(num1 + " + " + num2 + " = " + sum);
12    }
13 }
```

Output



The screenshot shows the 'Output - SumOfNumbers (run)' window. It contains the following text: 'run:', '74 + 36 = 110', and 'BUILD SUCCESSFUL (total time: 0 seconds)'. On the left side of the window, there are icons for running, debugging, and other IDE functions.

```
run:
74 + 36 = 110
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Write a Java program to divide two numbers and print them on the screen. Test Data :

50/3

Expected Output :

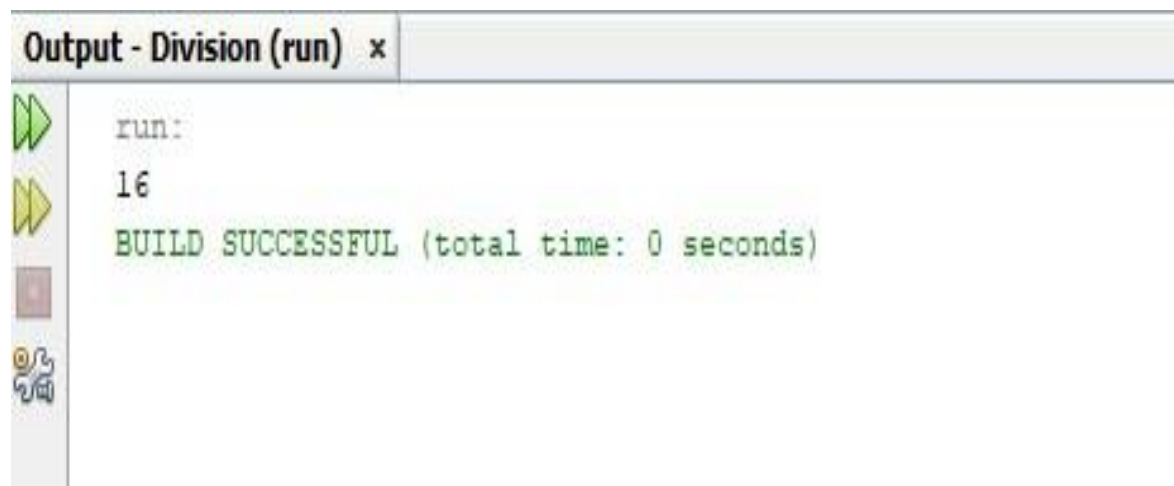
16

```

1 public class Division {
2     public static void main(String[] args) {
3         // Define the two numbers to be divided
4         int num1 = 50;
5         int num2 = 3;
6
7         // Calculate the division of the two numbers and print it
8         System.out.println(num1 / num2);
9     }
10 }

```

Output



```

run:
16
BUILD SUCCESSFUL (total time: 0 seconds)

```

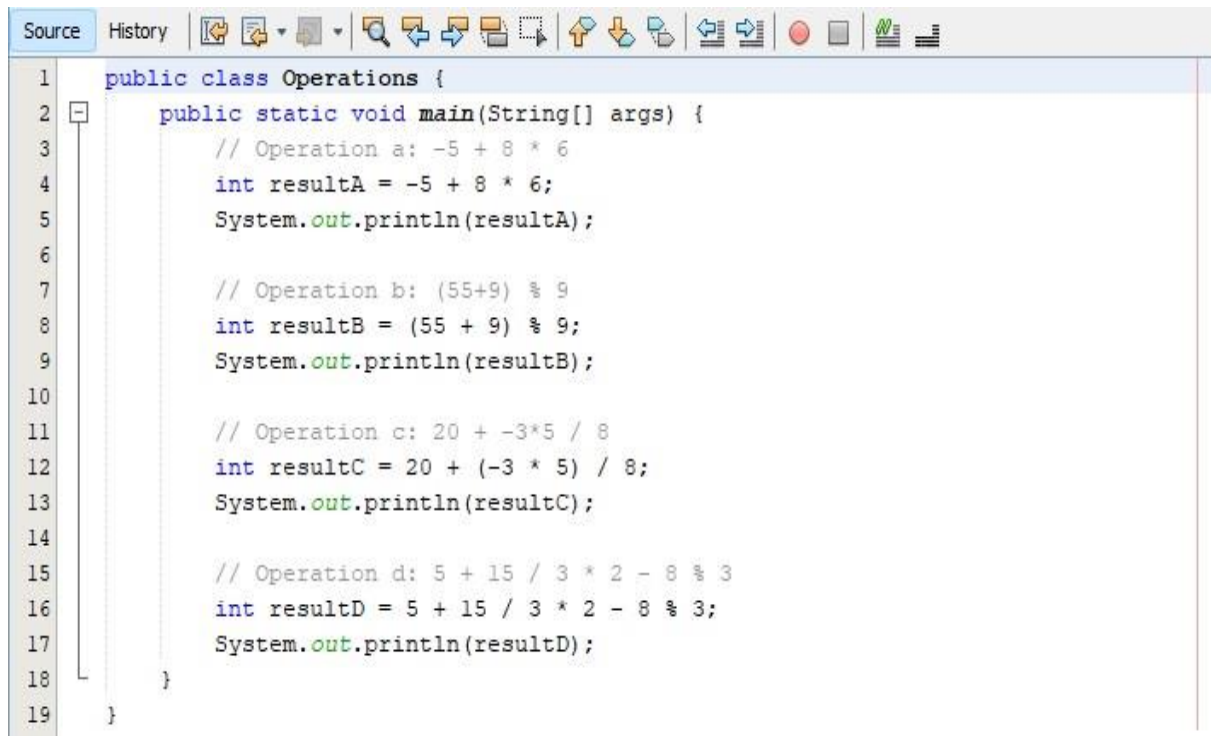
4. Write a Java program to print the results of the following operations. Test Data: a. -5 + 8 * 6 b. (55+9) % 9 c. 20 + -3*5 / 8 d. 5 + 15 / 3 * 2 - 8 % 3 Expected Output :

43

1

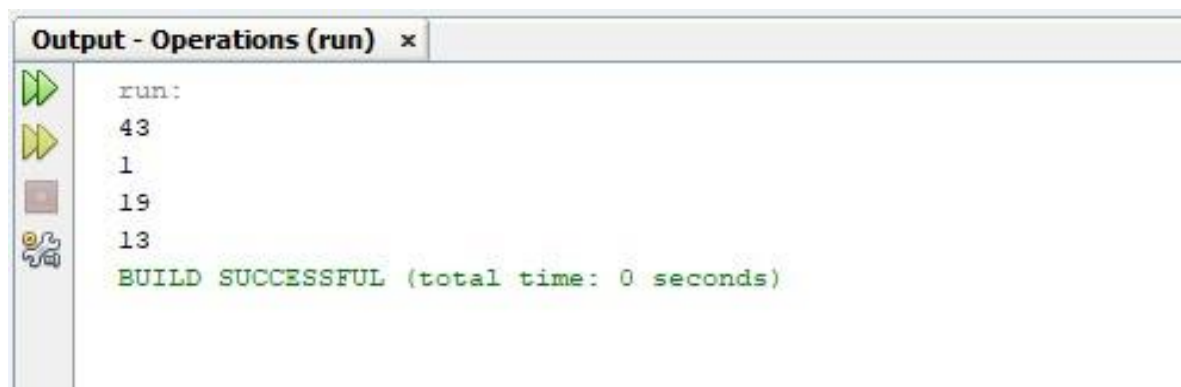
19

13



```
1 public class Operations {
2     public static void main(String[] args) {
3         // Operation a: -5 + 8 * 6
4         int resultA = -5 + 8 * 6;
5         System.out.println(resultA);
6
7         // Operation b: (55+9) % 9
8         int resultB = (55 + 9) % 9;
9         System.out.println(resultB);
10
11        // Operation c: 20 + -3*5 / 8
12        int resultC = 20 + (-3 * 5) / 8;
13        System.out.println(resultC);
14
15        // Operation d: 5 + 15 / 3 * 2 - 8 % 3
16        int resultD = 5 + 15 / 3 * 2 - 8 % 3;
17        System.out.println(resultD);
18    }
19 }
```

Output



```
run:
43
1
19
13
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Write a Java program that takes two numbers as input and displays the product of two numbers. Test Data: Input first number: 25 Input second number: 5 Expected Output : 25 x 5 = 125


```

1  package productcalculator;
2
3  import java.util.Scanner;
4
5  public class ProductCalculator {
6      public static void main(String[] args) {
7          // Create a Scanner object for user input
8          Scanner scanner = new Scanner(System.in);
9
10         // Ask the user to input the first number
11         System.out.print("Input first number: ");
12         int num1 = scanner.nextInt();
13
14         // Ask the user to input the second number
15         System.out.print("Input second number: ");
16         int num2 = scanner.nextInt();
17
18         // Close the scanner
19         scanner.close();

```

```

15         System.out.print("Input second number: ");
16         int num2 = scanner.nextInt();
17
18         // Close the scanner
19         scanner.close();
20
21         // Calculate the product of the two numbers
22         int product = num1 * num2;
23
24         // Display the result
25         System.out.println(num1 + " x " + num2 + " = " + product);
26     }
27 }

```

PART3

1. Write a Java program to create a base class Animal (Animal Family) with a method called Sound(). Create two subclasses Bird and Cat. Override the Sound() method in each subclass to make a specific sound for each animal.

```

1 // Define the base class Animal
2 class Animal {
3     // Method to make sound
4     public void sound() {
5         System.out.println("Animal makes a sound");
6     }
7 }
8
9 // Subclass Bird extending Animal
10 class Bird extends Animal {
11     // Override the sound method to make bird sound
12     public void sound() {
13         System.out.println("Bird chirps");
14     }
15 }
16
17 // Subclass Cat extending Animal
18 class Cat extends Animal {
19     // Override the sound method to make cat sound

```

```

19 // Override the sound method to make cat sound
20 @Override
21 public void sound() {
22     System.out.println("Cat meows");
23 }
24 }
25
26 public class Main {
27     public static void main(String[] args) {
28         // Create objects of Bird and Cat
29         Bird bird = new Bird();
30         Cat cat = new Cat();
31
32         // Call sound method for Bird and Cat
33         bird.sound();
34         cat.sound();
35     }
36 }
37

```

Output - Animal (run) x

```

run:
Bird chirps
Cat meows
BUILD SUCCESSFUL (total time: 0 seconds)

```





- Write a Java program to create a class Vehicle with a method called speedUp(). Create two subclasses Car and Bicycle. Override the speedUp() method in each subclass to increase the vehicle's speed differently.

```

1 // Base class Vehicle
2 class Vehicle {
3     public void speedUp() {
4         System.out.println("Increasing speed of the vehicle");
5     }
6 }
7
8 // Subclass Car
9 class Car extends Vehicle {
10    public void speedUp() {
11        System.out.println("Speeding up the car");
12    }
13 }
14
15 // Subclass Bicycle
16 class Bicycle extends Vehicle {
17    public void speedUp() {
18        System.out.println("Pedaling faster on the bicycle");
19    }
20 }
21
22 public class VehicleTest {
23     public static void main(String[] args) {
24         Vehicle car = new Car();
25         Vehicle bicycle = new Bicycle();
26
27         System.out.print("Car: ");
28         car.speedUp();
29
30         System.out.print("Bicycle: ");
31         bicycle.speedUp();
32     }
33 }

```

Output

Output - Vehicle (run) x	
	run:
	Car: Speeding up the car
	Bicycle: Pedaling faster on the bicycle
	BUILD SUCCESSFUL (total time: 0 seconds)

- Write a Java program to create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

```
1 // Define the base class Shape
2 class Shape {
3     // Method to calculate area
4     public double calculateArea() {
5         return 0; // Default implementation returns 0
6     }
7 }
8
9 // Subclass Circle extending Shape
10 class Circle extends Shape {
11     private double radius;
12
13     // Constructor to initialize radius
14     public Circle(double radius) {
15         this.radius = radius;
16     }
17
18     // Override the calculateArea method to calculate circle's area
19     @Override
```

```
Source History [Icons]
19 @Override
20 public double calculateArea() {
21     double area = Math.PI * radius * radius;
22     return Math.round(area * 10.0) / 10.0; // Round to 1 decimal place
23 }
24 }
25
26 // Subclass Rectangle extending Shape
27 class Rectangle extends Shape {
28     private double length;
29     private double width;
30
31     // Constructor to initialize length and width
32     public Rectangle(double length, double width) {
33         this.length = length;
34         this.width = width;
35     }
36
37     // Override the calculateArea method to calculate rectangle's area
```

```

37     // Override the calculateArea method to calculate rectangle's area
38     @Override
39     public double calculateArea() {
40         return length * width;
41     }
42 }
43
44 // Subclass Triangle extending Shape
45 class Triangle extends Shape {
46     private double base;
47     private double height;
48
49     // Constructor to initialize base and height
50     public Triangle(double base, double height) {
51         this.base = base;
52         this.height = height;
53     }
54
55     // Override the calculateArea method to calculate triangle's area

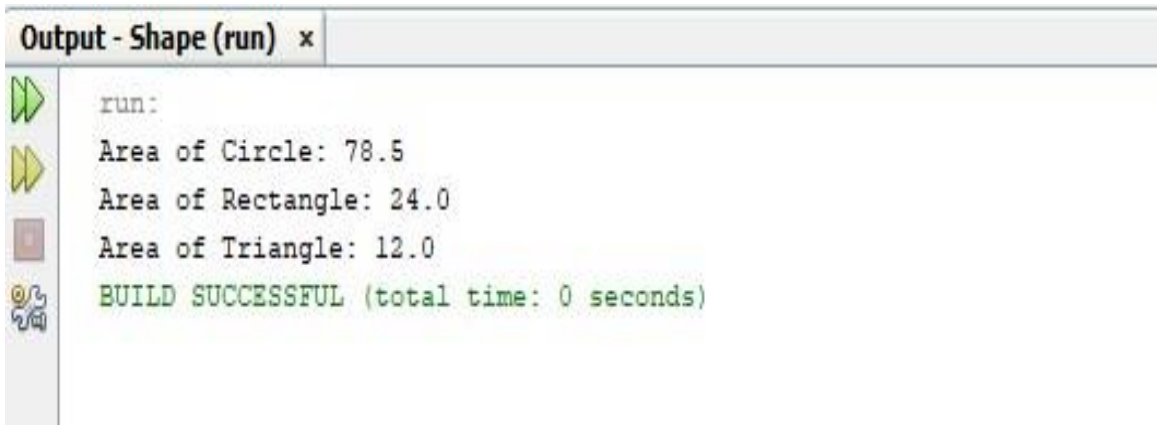
```

```

56     @Override
57     public double calculateArea() {
58         return 0.5 * base * height;
59     }
60 }
61
62 public class Main {
63     public static void main(String[] args) {
64         // Create objects of Circle, Rectangle, and Triangle
65         Circle circle = new Circle(5);
66         Rectangle rectangle = new Rectangle(4, 6);
67         Triangle triangle = new Triangle(3, 8);
68
69         // Calculate and print the area of each shape
70         System.out.println("Area of Circle: " + circle.calculateArea());
71         System.out.println("Area of Rectangle: " + rectangle.calculateArea());
72         System.out.println("Area of Triangle: " + triangle.calculateArea());
73     }
74 }

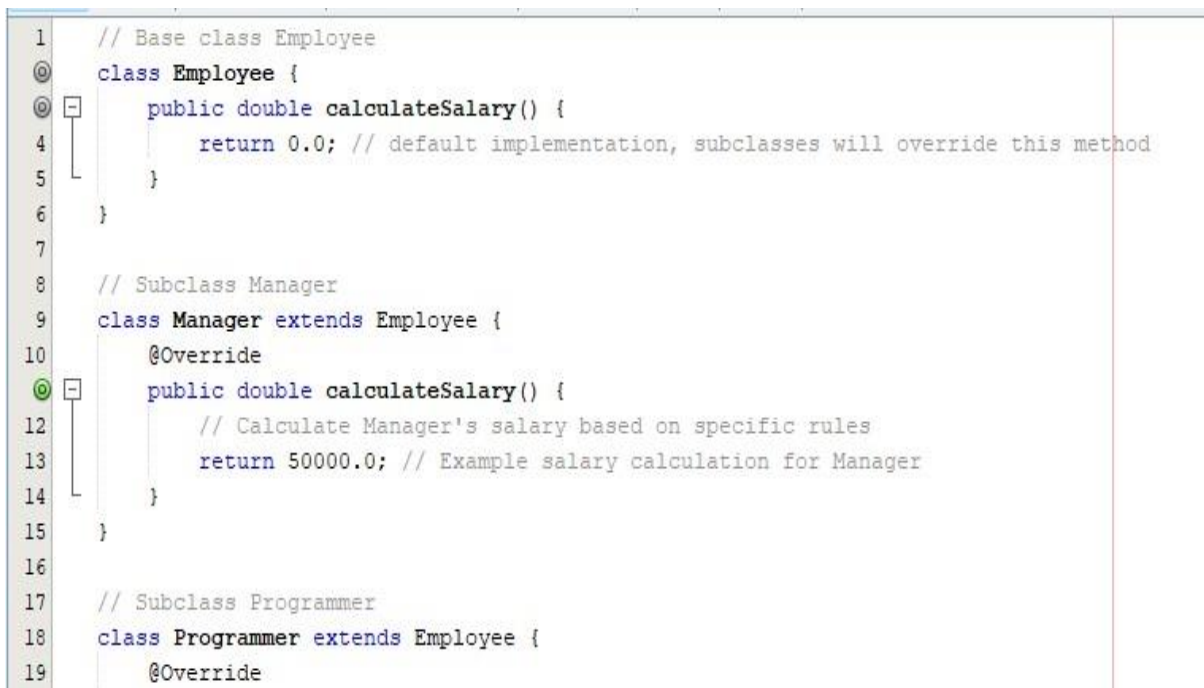
```

Output



```
Output - Shape (run) x
run:
Area of Circle: 78.5
Area of Rectangle: 24.0
Area of Triangle: 12.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Write a Java program to create a class Employee with a method called calculateSalary(). Create two subclasses Manager and Programmer. In each subclass, override the calculateSalary() method to calculate and return the salary based on their specific roles



```
1 // Base class Employee
2 class Employee {
3     public double calculateSalary() {
4         return 0.0; // default implementation, subclasses will override this method
5     }
6 }
7
8 // Subclass Manager
9 class Manager extends Employee {
10     @Override
11     public double calculateSalary() {
12         // Calculate Manager's salary based on specific rules
13         return 50000.0; // Example salary calculation for Manager
14     }
15 }
16
17 // Subclass Programmer
18 class Programmer extends Employee {
19     @Override
```

```

16
17 // Subclass Programmer
18 class Programmer extends Employee {
19     @Override
20     public double calculateSalary() {
21         // Calculate Programmer's salary based on specific rules
22         return 60000.0; // Example salary calculation for Programmer
23     }
24 }
25
26 public class EmployeeTest {
27     public static void main(String[] args) {
28         Manager manager = new Manager();
29         Programmer programmer = new Programmer();
30
31         System.out.println("Manager's Salary: $" + manager.calculateSalary());
32         System.out.println("Programmer's Salary: $" + programmer.calculateSalary());
33     }
34 }

```

Output

```

Output - Employee (run) x
run:
Manager's Salary: $50000.0
Programmer's Salary: $60000.0
BUILD SUCCESSFUL (total time: 1 second)

```

- Write a Java program to create a base class Sports with a method called play(). Create three subclasses: Football, Basketball, and Rugby. Override the play() method in each subclass to play a specific statement for each sport.

```

1 // Base class Sports
2 class Sports {
3     public void play() {
4         System.out.println("Let's play a sport!");
5     }
6 }
7
8 // Subclass Football
9 class Football extends Sports {
10     @Override
11     public void play() {
12         System.out.println("Let's play Football!");
13     }
14 }
15
16 // Subclass Basketball
17 class Basketball extends Sports {
18     @Override
19     public void play() {

```



```

20     public void play() {
21         System.out.println("Let's play Basketball!");
22     }
23
24     // Subclass Rugby
25     class Rugby extends Sports {
26         @Override
27         public void play() {
28             System.out.println("Let's play Rugby!");
29         }
30     }
31
32     public class SportsTest {
33         public static void main(String[] args) {
34             Football football = new Football();
35             Basketball basketball = new Basketball();
36             Rugby rugby = new Rugby();
37

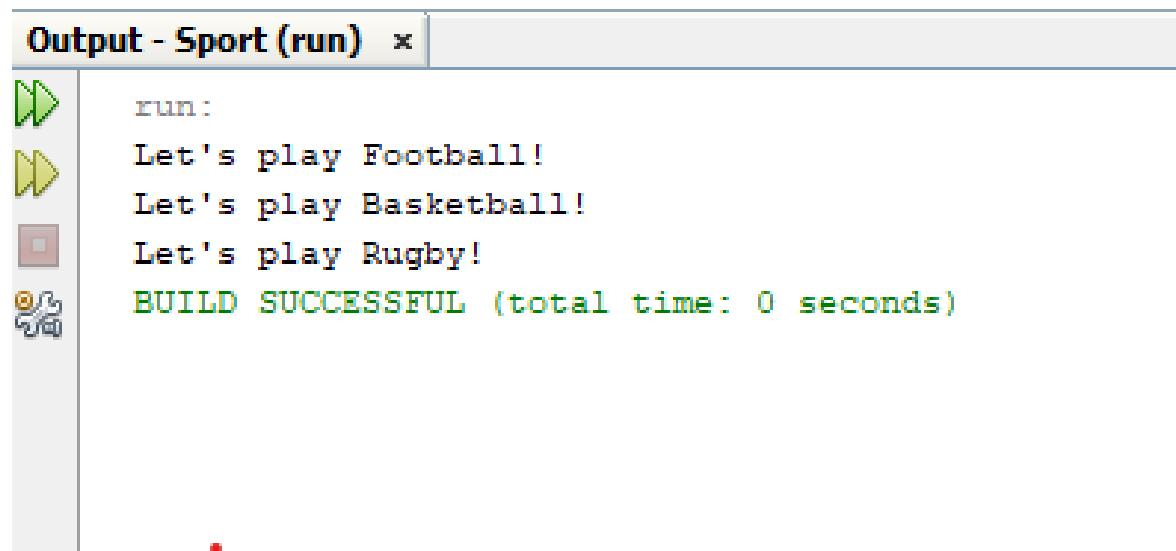
```

```

30     }
31
32     public class SportsTest {
33         public static void main(String[] args) {
34             Football football = new Football();
35             Basketball basketball = new Basketball();
36             Rugby rugby = new Rugby();
37
38             football.play();
39             basketball.play();
40             rugby.play();
41         }
42     }

```

Output



```
run:
Let's play Football!
Let's play Basketball!
Let's play Rugby!
BUILD SUCCESSFUL (total time: 0 seconds)
```

The End