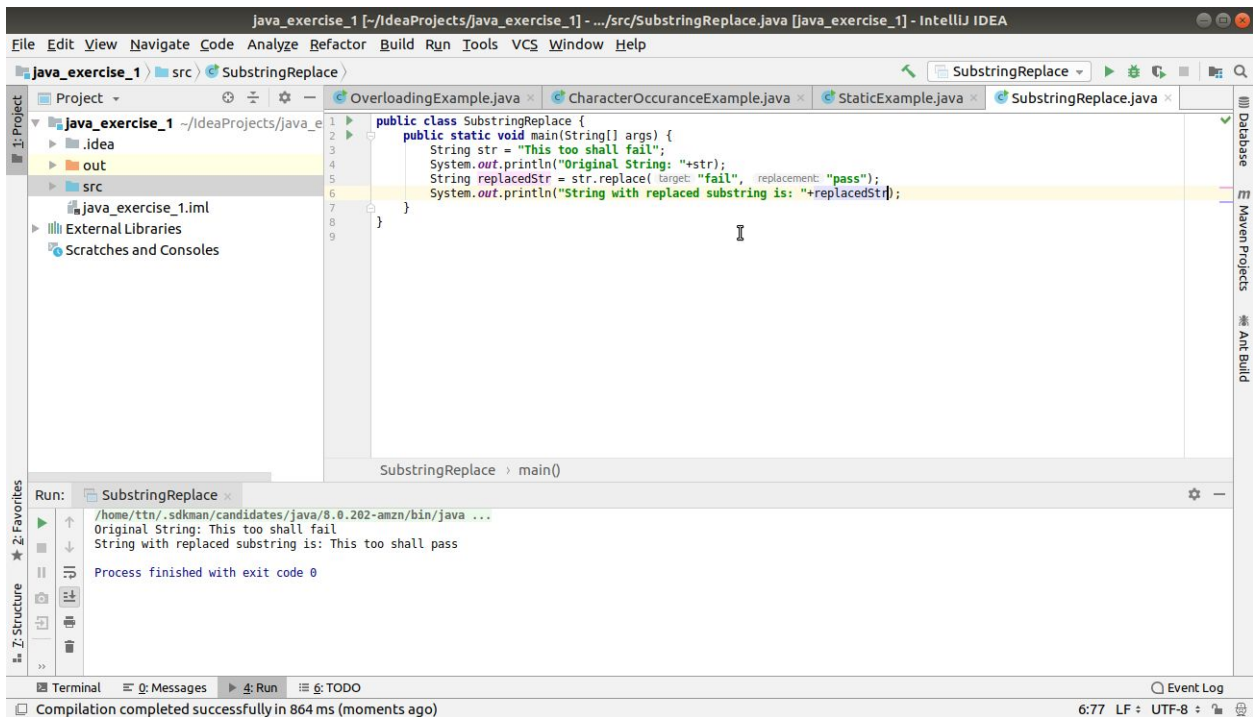


JAVA ASSIGNMENT I

Q.1. Write a program to replace a substring inside a string with other string ?



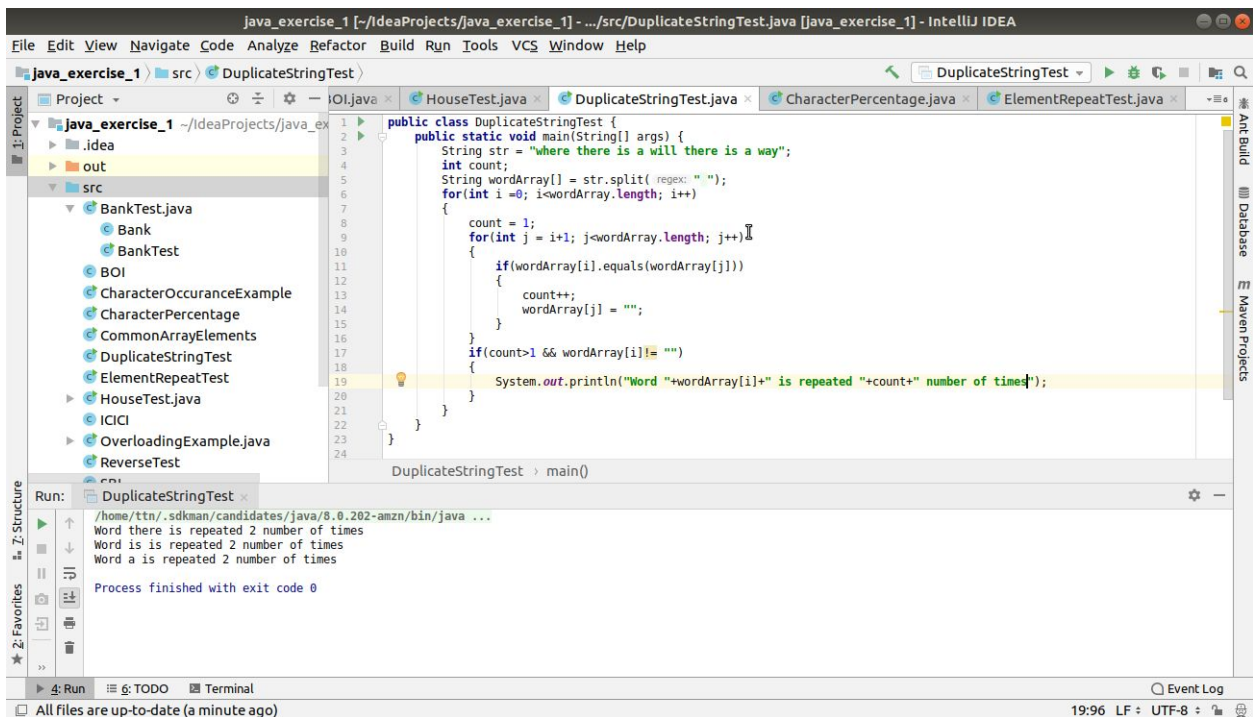
The screenshot shows the IntelliJ IDEA interface with the file `SubstringReplace.java` open. The code defines a `SubstringReplace` class with a `main` method that takes an array of arguments. It initializes a string `str = "This too shall fail";`, prints the original string, replaces the substring `"fail"` with `"pass"` using `str.replace(target, replacement)`, and prints the resulting string `"String with replaced substring is: " + replacedStr`.

```
1 public class SubstringReplace {
2     public static void main(String[] args) {
3         String str = "This too shall fail";
4         System.out.println("Original String: "+str);
5         String replacedStr = str.replace(target: "fail", replacement: "pass");
6         System.out.println("String with replaced substring is: "+replacedStr);
7     }
8 }
9
```

The Run window shows the output of the program:

```
Run: SubstringReplace
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Original String: This too shall fail
String with replaced substring is: This too shall pass
Process finished with exit code 0
```

Q.2. Write a program to find the number of occurrences of the duplicate words in a string and print them ?



The screenshot shows the IntelliJ IDEA interface with the file `DuplicateStringTest.java` open. The code defines a `DuplicateStringTest` class with a `main` method that takes an array of arguments. It initializes a string `str = "where there is a will there is a way";`, splits it into an array of words using `str.split(regex: " ")`, and iterates through the array to find duplicate words. It uses a nested loop to compare each word with the subsequent words in the array. If a duplicate is found, it increments the count and prints the word and its count.

```
1 public class DuplicateStringTest {
2     public static void main(String[] args) {
3         String str = "where there is a will there is a way";
4         int count;
5         String wordArray[] = str.split(regex: " ");
6         for(int i=0; i<wordArray.length; i++)
7         {
8             count = 1;
9             for(int j = i+1; j<wordArray.length; j++)
10            {
11                if(wordArray[i].equals(wordArray[j]))
12                {
13                    count++;
14                    wordArray[j] = "";
15                }
16            }
17            if(count>1 && wordArray[i]!="")
18            {
19                System.out.println("Word "+wordArray[i]+" is repeated "+count+" number of times");
20            }
21        }
22    }
23 }
24
```

The Run window shows the output of the program:

```
Run: DuplicateStringTest
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Word there is repeated 2 number of times
Word is is repeated 2 number of times
Word a is repeated 2 number of times
Process finished with exit code 0
```

Q3. Write a program to find the number of occurrences of a character in a string without using loop?

The screenshot shows the IntelliJ IDEA interface with the `CharacterOccuranceExample.java` file open. The code defines a public class `CharacterOccuranceExample` with a `main` method that takes an array of strings as input. It processes the first string, `"Occasion"`, by replacing all occurrences of the character 'c' with an empty string. It then calculates the difference between the original length and the new length to find the number of occurrences.

```

1 public class CharacterOccuranceExample {
2     public static void main(String[] args) {
3         String str = "Occasion";
4         int len = str.length();
5         String replaced_str = str.replaceAll("c", "");
6         int replaced_length = replaced_str.length();
7         int difference = len - replaced_length;
8         System.out.println("the number of times 'c' occured is "+difference);
9     }
10 }
11
12
13

```

The Run window shows the output: `the number of times 'c' occured is 2`. The process finished with exit code 0.

Q4. Calculate the number & Percentage Of Lowercase Letters,Uppercase Letters, Digits And Other Special Characters In A String

The screenshot shows the IntelliJ IDEA interface with the `CharacterPercentage.java` file open. The code defines a public class `CharacterPercentage` with a `main` method that takes an array of strings as input. It processes the first string, `"QSDHJifhcdsb +1325fg"`, by iterating through each character and counting the number of uppercase letters, lowercase letters, digits, and special characters. It then calculates the percentage for each category.

```

1 public class CharacterPercentage {
2     public static void main(String[] args) {
3         String str = "QSDHJifhcdsb +1325fg";
4         int uppCase = 0, lowCase = 0, digits = 0, spcCharacters = 0, len = str.length();
5         for(int k=0; k < str.length(); k++)
6         {
7             //uppercase
8             if(Character.isUpperCase(str.charAt(k)))
9             {
10                 uppCase++;
11             }
12             //lowercase
13             if(Character.isLowerCase(str.charAt(k)))
14             {
15                 lowCase++;
16             }
17             // digits
18             if(Character.isDigit(str.charAt(k)))
19             {
20                 digits++;
21             }
22             if (!Character.isDigit(str.charAt(k)) && !Character.isAlphabetic(str.charAt(k)))
23             {
24                 spcCharacters++;
25             }
26         }
27         System.out.println("the number of uppercase characters are 5 with % equal to 22.727274");
28         System.out.println("the number of lowercase characters are 11 with % equal to 50.0");
29         System.out.println("the number of digits are 4 with % equal to 18.1818");
30         System.out.println("the number of special characters are 2 with % equal to 9.090909");
31     }
32 }
33
34
35

```

The Run window shows the output: `the number of uppercase characters are 5 with % equal to 22.727274`, `the number of lowercase characters are 11 with % equal to 50.0`, `the number of digits are 4 with % equal to 18.1818`, and `the number of special characters are 2 with % equal to 9.090909`. The process finished with exit code 0.

Q5. Find common elements between two arrays.

The screenshot shows the IntelliJ IDEA interface with the `CommonArrayElements.java` file open. The code defines a `main` method that takes two integer arrays, `array1` and `array2`, and finds their common elements. The arrays are `{1, 2, 3, 4, 5}` and `{4, 5, 6, 7, 8, 1, 3}`. The code uses nested loops to compare elements and prints the common elements to the console.

```
1 public class CommonArrayElements {
2     public static void main(String[] args) {
3         int array1[] = {1, 2, 3, 4, 5};
4         int array2[] = {4, 5, 6, 7, 8, 1, 3};
5         int len1 = array1.length;
6         int len2 = array2.length;
7         for(int i = 0; i < len1; i++)
8         {
9             for(int j = 0; j < len2; j++)
10            {
11                if(array1[i] == array2[j])
12                {
13                    // printing common elements
14                    System.out.println(array1[i]);
15                }
16            }
17        }
18    }
19 }
20
```

The Run window shows the output of the program:

```
1
2
3
4
5
```

The status bar at the bottom indicates that all files are up-to-date.

Q6. There is an array with every element repeated twice except one. Find that element

The screenshot shows the IntelliJ IDEA interface with the `ElementRepeatTest.java` file open. The code defines a `findSingle` method that uses the XOR operation to find the single element in an array where every other element is repeated twice. The array is `{1, 4, 3, 4, 1}`. The code prints the result to the console.

```
1 public class ElementRepeatTest {
2     static void findSingle(int array[], int arraySize)
3     {
4         int resultXOR = array[0];
5         for (int i = 1; i < arraySize; i++)
6         {
7             resultXOR = resultXOR ^ array[i];
8         }
9         System.out.println(resultXOR);
10        return resultXOR;
11    }
12    //
13
14    public static void main(String[] args) {
15        int dummyArray[] = {1, 4, 3, 4, 1};
16        int n = dummyArray.length;
17        System.out.println("Element occurring twice is: ");
18        findSingle(dummyArray, dummyArray.length);
19    }
20 }
21
22
23
```

The Run window shows the output of the program:

```
Element occurring twice is:
3
```

The status bar at the bottom indicates that the compilation completed successfully in 770 ms.

Q7. Write a program to print your Firstname, LastName & age using static block, static method & static variable respectively

The screenshot shows the IntelliJ IDEA interface with the file `StaticExample.java` open. The code defines a class `StaticExample` with static variables `firstName`, `lastName`, and `age`, a static block, a static method `myMethod`, and a `main` method.

```

class StaticExample {
    static String firstName = "Divya";
    static String lastName = "Arora";
    static int age = 23;

    static {
        System.out.println("Firstname: "+firstName);
    }

    static void myMethod() {
        System.out.println("Lastname: "+lastName);
    }

    void print_age() {
        System.out.println("Age: "+age);
    }

    public static void main(String[] args) {
        StaticExample.myMethod();
        System.out.println(StaticExample.age);
    }
}

```

The Run window shows the output of the program:

```

/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Firstname: Divya
Lastname: Arora
23
Process finished with exit code 0

```

The status bar at the bottom indicates "Compilation completed successfully in 928 ms (a minute ago)".

Q8. Write a program to reverse a string and remove character from index 4 to index 9 from the reversed string using String Buffer

The screenshot shows the IntelliJ IDEA interface with the file `ReverseTest.java` open. The code defines a class `ReverseTest` with a `main` method that reverses a string and then removes characters from index 4 to index 9 using a `StringBuffer`.

```

public class ReverseTest {
    public static void main(String[] args) {
        String str = "Dummy String";
        char ch[] = str.toCharArray();
        String reverseStr = "";
        for (int i = ch.length-1; i>=0; i--) {
            reverseStr = reverseStr+ch[i];
        }
        System.out.println("The Reversed String is: "+reverseStr);
        StringBuffer sb = new StringBuffer(reverseStr);
        sb.delete(4, 9);
        System.out.println("Final String: "+sb);
    }
}

```

The Run window shows the output of the program:

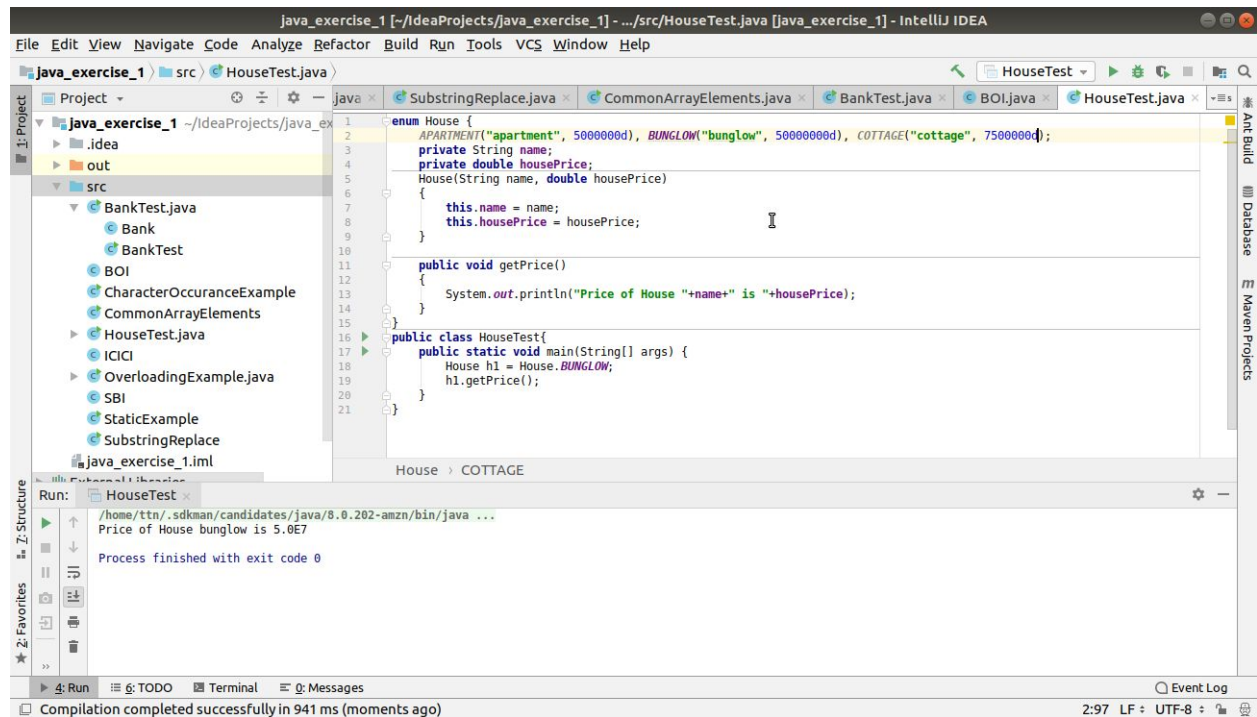
```

/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
The Reversed String is: gnirts ymmuD
Final String: gnirmuD
Process finished with exit code 0

```

The status bar at the bottom indicates "Compilation completed successfully in 854 ms (moments ago)".

Q9. Write a program to display values of enums using a constructor & getPrice() method (Example display house & their prices)



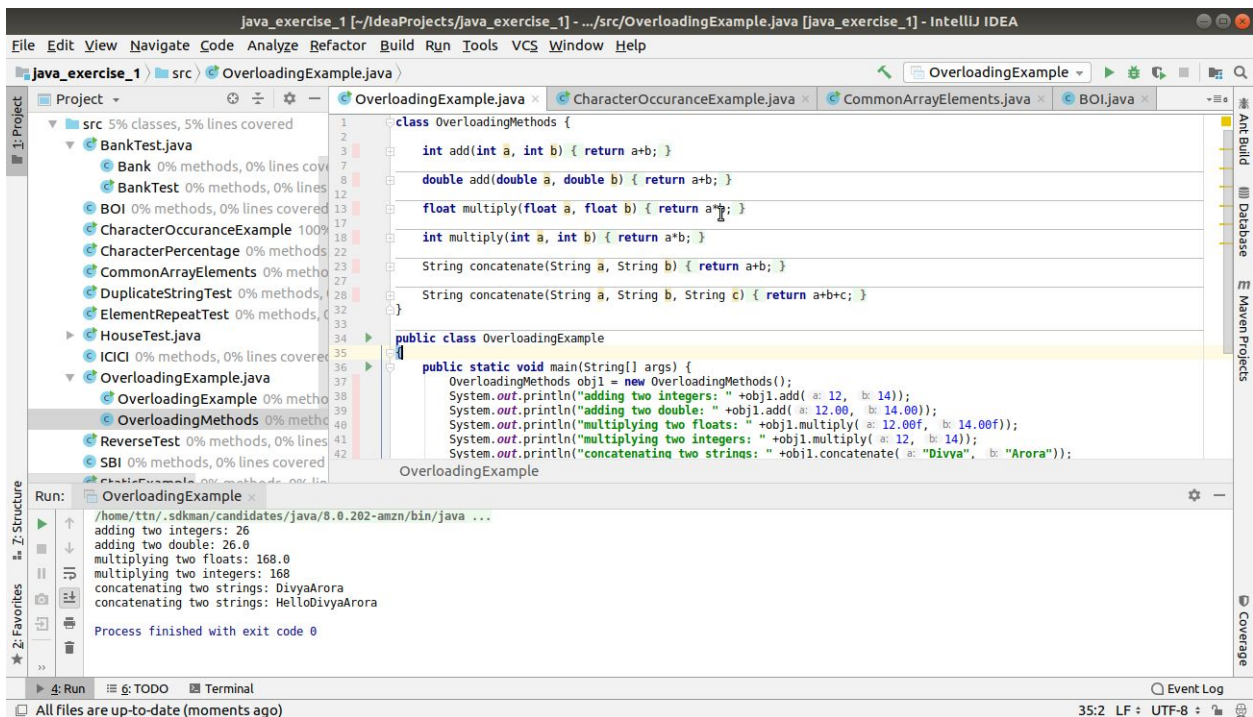
The screenshot shows the IntelliJ IDEA IDE with a project named 'java_exercise_1'. The main editor displays the 'HouseTest.java' file. The code defines an enum 'House' with three values: 'APARTMENT', 'BUNGLOW', and 'COTTAGE', each with a name and a price. A constructor 'House(String name, double housePrice)' initializes the 'name' and 'housePrice' fields. A 'getPrice()' method is also defined. The 'HouseTest' class contains a 'main' method that creates a 'BUNGLOW' object and calls 'getPrice()' to display the price.

```
1 enum House {  
2     APARTMENT("apartment", 5000000d), BUNGLOW("bungalow", 50000000d), COTTAGE("cottage", 7500000d);  
3     private String name;  
4     private double housePrice;  
5     House(String name, double housePrice)  
6     {  
7         this.name = name;  
8         this.housePrice = housePrice;  
9     }  
10  
11     public void getPrice()  
12     {  
13         System.out.println("Price of House "+name+" is "+housePrice);  
14     }  
15 }  
16  
17 public class HouseTest {  
18     public static void main(String[] args) {  
19         House h1 = House.BUNGLOW;  
20         h1.getPrice();  
21     }  
22 }
```

The 'Run' tab at the bottom shows the output: 'Price of House bungalow is 5.0E7' and 'Process finished with exit code 0'.

Q10. Write a single program for following operation using overloading

- A) Adding 2 integer number**
- B) Adding 2 double**
- C) multiplying 2 float**
- D) multiplying 2 int**
- E) concate 2 string**
- F) Concate 3 String**



```
java_exercise_1 [-/IdeaProjects/java_exercise_1] - .../src/OverloadingExample.java [java_exercise_1] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project: java_exercise_1 | src | OverloadingExample.java | OverloadingExample | CharacterOccuranceExample.java | CommonArrayElements.java | BOI.java | ...
src 5% classes, 5% lines covered
  BankTest.java
  BankTest 0% methods, 0% lines covered
  BOI 0% methods, 0% lines covered
  CharacterOccuranceExample 100% methods, 100% lines covered
  CharacterPercentage 0% methods, 0% lines covered
  CommonArrayElements 0% methods, 0% lines covered
  DuplicateStringTest 0% methods, 0% lines covered
  ElementRepeatTest 0% methods, 0% lines covered
  HouseTest.java
  ICICI 0% methods, 0% lines covered
  OverloadingExample.java
  OverloadingExample 0% methods, 0% lines covered
  OverloadingMethods 0% methods, 0% lines covered
  ReverseTest 0% methods, 0% lines covered
  SBI 0% methods, 0% lines covered

1 class OverloadingMethods {
2     int add(int a, int b) { return a+b; }
3
4     double add(double a, double b) { return a+b; }
5
6     float multiply(float a, float b) { return a*b; }
7
8     int multiply(int a, int b) { return a*b; }
9
10    String concatenate(String a, String b) { return a+b; }
11
12    String concatenate(String a, String b, String c) { return a+b+c; }
13
14    public class OverloadingExample
15    {
16        public static void main(String[] args) {
17            OverloadingMethods obj1 = new OverloadingMethods();
18            System.out.println("adding two integers: " + obj1.add(12, 14));
19            System.out.println("adding two double: " + obj1.add(12.00, 14.00));
20            System.out.println("multiplying two floats: " + obj1.multiply(12.00f, 14.00f));
21            System.out.println("multiplying two integers: " + obj1.multiply(12, 14));
22            System.out.println("concatenating two strings: " + obj1.concatenate("Divya", "Arora"));
23        }
24    }
25}
```

Run: OverloadingExample x

```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
adding two integers: 26
adding two double: 26.0
multiplying two floats: 168.0
multiplying two integers: 168
concatenating two strings: DivyaArora
concatenating two strings: HelloDivyaArora

Process finished with exit code 0
```

Run | TODO | Terminal | Event Log

All files are up-to-date (moments ago) 35:2 LF : UTF-8

Q11.Create 3 sub class of bank SBI,BOI,ICICI all 4 should have method called getDetails which provide there specific details like rateofinterest etc,print details of every banks

```

1 class Bank {
2     String bankName;
3     double rateOfInterest;
4
5     Bank(String bankName, double rateOfInterest) {
6         this.bankName = bankName;
7         this.rateOfInterest = rateOfInterest;
8     }
9
10    void getDetails() { System.out.println("Greetings from the Bank: "); }
11
12    public class BankTest {
13        public static void main(String[] args) {
14            Bank b1 = new ICICI( "bankName: \"ICICI\", rateOfInterest: 12);
15            Bank b2 = new SBI( "bankName: \"SBI\", rateOfInterest: 7);
16            Bank b3 = new BOI( "bankName: \"BOI\", rateOfInterest: 10);
17
18            b1.getDetails();
19            b2.getDetails();
20            b3.getDetails();
21        }
22    }
23 }

```

Run: BankTest

```

/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Greetings from the Bank:
Bank Name: ICICI
Rate of interest: 12.0
Greetings from the Bank:
Bnk name: SBI
Rate of Interest: 7.0
Greetings from the Bank:
Bank NameBOI
Rate of Interest is: 10.0
Process finished with exit code 0

```

```

1 public class ICICI extends Bank {
2     ICICI(String bankName, double rateOfInterest) {
3         super(bankName, rateOfInterest);
4     }
5
6     @Override
7     void getDetails() {
8         super.getDetails();
9         System.out.println("Bank Name: "+bankName);
10        System.out.println("Rate of interest: "+rateOfInterest);
11    }
12 }

```

Run: BankTest

```

Greetings from the Bank:
Bank Name: ICICI
Rate of interest: 12.0
Greetings from the Bank:
Bnk name: SBI
Rate of Interest: 7.0
Greetings from the Bank:
Bank NameBOI
Rate of Interest is: 10.0
Process finished with exit code 0

```