

JAVA (Basics and Fundamentals) Assignment AU

In-Slide Questions

Question 1: Write a Java program to print the result of the following operations.

- a) $-5 + 8 * 6$
- c) $20 + -3*5 / 8$

- b) $(55+9) \% 9$
- d) $5 + 15 / 3*2-8 \% 3$

Solution:

```
2 import java.util.*;
3 //=====
4 //=====
5 //===== ACCOLITE UNIVERSITY JAVA ASSIGNMENT=====
6 //=====
7
8 public class Question1
9 {
10    public static void main(String[] args)
11    {
12        Scanner inp = new Scanner(System.in);
13        System.out.println("Enter the number of Expressions");
14        int n= inp.nextInt();
15        for(int co=0;co<n;co++) {
16            Stack<Integer> op = new Stack<Integer>();
17            Stack<Double> val = new Stack<Double>();
18            Stack<Integer> optmp = new Stack<Integer>();
19            Stack<Double> valtmp = new Stack<Double>();
20            System.out.println("\nEnter expression\n");
21            String input = inp.next();
22            input = "0" + input;
23            input = input.replaceAll("-","+-");
24            String temp = "";
25            for (int i = 0;i < input.length();i++)
26            {
27                char ch = input.charAt(i);
28                if (ch == '-')
29                    temp = "-"+ temp;
30                else if (ch != '+' && ch != '*' && ch != '/')
31                    temp = temp + ch;
32                else
33                {
34                    val.push(Double.parseDouble(temp));
35                    op.push((int)ch);
36                    temp = "";
37                }
38            }
39            val.push(Double.parseDouble(temp));
40            char operators[] = {'/','*','+'};
```

Figure 1 Code Snippet for Question1 (in slide question)

Question 2: Write a Java program to print an American flag on the screen.

Solution:

American flag can be broken into two patterns, one which contain repeating star pattern and other that contains repeating line pattern. When printed the right number of times, an American flag is rendered on the console output.

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows tabs for module-info.java, Question1.java, and Question2.java.
- Code Editor (Question2.java):**

```
1 package inSlideQuestions;
2
3 //===== ACCOLITE UNIVERSITY JAVA ASSIGNMENT=====
4 //-
5 //===== VISHNU SINGH SENGAR=====
6 //-
7
8 public class Question2 {
9
10    public static void main(String[] args) {
11        String patStars="* * * * *\n * * * * * =====";
12        String patLines="=====";
13        for(int i=0;i<4;i++)
14            System.out.println(patStars);
15        for(int i=0;i<6;i++)
16            System.out.println(patLines);
17    }
18
19
20 }
```
- Console Output:**

```
* * * * *\n =====\n =====\n =====\n =====\n =====\n =====\n =====\n =====
```

Figure 2 Code and Output for Question 2 (In Slide Assignment)

Hands On Questions

Question 1: String

Write a Java program to replace each substring of a given string that matches the given regular expression with the given replacement.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows two files: module-info.java and Question1.java.
- Code Editor:** Displays the Java code for Question 1. The code reads an original string from standard input, prompts for a word to be replaced, reads the replacement word, and then prints the modified string.
- Console Output:** Shows the execution of the program. It asks for the original string ("Enter the original string:"), prints "Accolite is leading company in service sector. All clients love Accolite.", asks for the word to be replaced ("Enter the word to be replaced:"), replaces "Accolite" with "Accolite Digital" ("Accolite"), asks for the replacement word ("Enter the replacement word:"), and finally prints the modified string ("Original String: Accolite is leading company in service sector. All clients love Accolite." and "New String: Accolite Digital is leading company in service sector. All clients love Accolite Digital.").

```
1 package handsOnQuestions;
2 import java.util.*;
3
4 public class Question1 {
5
6     public static void main(String[] args) {
7         Scanner inp=new Scanner(System.in);
8
9         System.out.println("Enter the original string: ");
10        String oriString=inp.nextLine();
11
12        System.out.println("Enter the word to be replaced: ");
13        String regEx=inp.nextLine();
14
15        System.out.println("Enter the replacement word: ");
16        String tarWord=inp.nextLine();
17
18        String newStr=oriString.replaceAll(regEx, tarWord);
19
20        System.out.println("Original String: "+oriString);
21        System.out.println("New String: "+newStr);
22    }
23
24 }
```

Problems @ Javadoc Declaration Console Progress
<terminated> Question1 (1) [Java Application] C:\Users\vishn\AppData\Local\Temp\eo146CF.tmp\plugins\org.eclipse.justj.openjdk.hotspot.jre

Enter the original string:
Accolite is leading company in service sector. All clients love Accolite.
Enter the word to be replaced:
Accolite
Enter the replacement word:
Accolite Digital
Original String: Accolite is leading company in service sector. All clients love Accolite.
New String: Accolite Digital is leading company in service sector. All clients love Accolite Digital.

Figure 3 Code + Output for Question 1 (Hands On)

Question 2: Collection

Write a Java program to get a reverse order view of the keys contained in a given map.

Solution:

The screenshot shows the Eclipse IDE interface with two tabs open: "Question2.java" and "Question1.java". The "Question2.java" tab contains the following Java code:

```
1 package handsOnQuestions;
2 import java.util.*;
3
4 public class Question2 {
5
6     public static void main(String[] args) {
7         TreeMap<String, String> dict = new TreeMap<>();
8         Scanner inp = new Scanner(System.in);
9
10        System.out.println("Enter the number of key->value pairs");
11        int n = inp.nextInt();
12
13        for(int i=0; i < n; i++) {
14            System.out.println("Enter Key");
15            String key = inp.next();
16            System.out.println("Enter Value");
17            String val = inp.next();
18            dict.put(key, val);
19        }
20        System.out.println("Original Map: " + dict);
21        System.out.println("Reversed Map: " + dict.descendingKeySet());
22    }
23 }
```

The "Console" tab shows the execution of the program and its output:

```
<terminated> Question2 (1) [Java Application] C:\Users\wishn\AppData\Local\Temp\eo146CF.tmp\plugins\org.eclipse.jdt.core\src\Question2.java
Enter the number of key->value pairs
5
Enter Key
Morgan
Enter Value
Accolite
Enter Key
FedEx
Enter Value
Accolite
Enter Key
MountainCapital
Enter Value
Accolite
Enter Key
Delloitte
Enter Value
Accolite
Enter Key
Wave
Enter Value
Accolite
Original Map: {Delloitte=Accolite, FedEx=Accolite, Morgan=Accolite, MountainCapital=Accolite, Wave=Accolite}
Reversed Map: [Wave, MountainCapital, Morgan, FedEx, Delloitte]
```

Figure 4 Code + Output for Question 2 (HandsOn)

Question 3: Exception

Write your own unchecked Exception and throw it from you counter programme which counts 1 to 100. When you get Prime no while counting then throw this Exception and catch this to print you exception message.

Solution:

The screenshot shows an IDE interface with several tabs at the top: module-info.java, Question2.java, Question1.java, and Question3.java. The Question3.java tab is active, displaying the following Java code:

```
1 package handsOnQuestions;
2 import java.util.*;
3
4 class ExceptPrime extends Exception{
5
6 }
7
8 public class Question3 {
9     public static boolean checkPrime(int n) {
10         if(n<1)
11             return false;
12         if(n<=3)
13             return true;
14         if(n%2==0||n%3==0)
15             return false;
16         for(int i=5;i*i<=n;i+=6)
17             if(n%i==0 || n%(i+2)==0)
18                 return false;
19         return true;
20     }
21
22     public static void main(String[] args) {
23         int count=1;
24         while(count<=100) {
25             try {
26                 if(checkPrime(count))
27                     throw new ExceptPrime();
28             }
29             catch(ExceptPrime e) {
30                 System.out.println(count+" is a Prime Number");
31             }
32             finally {
33                 count++;
34             }
35         }
36     }
37
38
39 }
```

To the right of the code editor is a 'Console' window showing the output of the program. The output lists all prime numbers from 2 to 97, each preceded by the message "is a Prime Number".

Output Message	Value
is a Prime Number	2
is a Prime Number	3
is a Prime Number	5
is a Prime Number	7
is a Prime Number	11
is a Prime Number	13
is a Prime Number	17
is a Prime Number	19
is a Prime Number	23
is a Prime Number	29
is a Prime Number	31
is a Prime Number	37
is a Prime Number	41
is a Prime Number	43
is a Prime Number	47
is a Prime Number	53
is a Prime Number	59
is a Prime Number	61
is a Prime Number	67
is a Prime Number	71
is a Prime Number	73
is a Prime Number	79
is a Prime Number	83
is a Prime Number	89
is a Prime Number	97

Figure 5 Code + Output for Question 3 (Hands On)

Question 4: Serialization

Write a programme to serialize 3 fields out of 5 and deserialize it. Use UUID to prevent object mutation.

Solution:

The screenshot shows a Java application running in an IDE. On the left, the code for `Question4.java` is displayed. On the right, the application's window shows the output of the program, which includes sections for serializing, deserializing, and data after deserialization.

```
1 import java.util.*;
2
3 class Data implements Serializable {
4     UUID serialVersionUID = new UUID(100,10);
5     transient int id;
6     transient int age;
7     String name;
8
9
10    public Data(int id, String name, int age)
11    {
12        this.id = id;
13        this.name = name;
14        this.age = age;
15    }
16
17 }
18
19 public class Question4{
20    public static void display(Data d){
21        System.out.println("ID Data: " + d.id);
22        System.out.println("Name Data: " + d.name);
23        System.out.println("Age Data: " + d.age);
24    }
25
26    public static void main(String []args){
27        Data d1 = new Data(1, "Vishnu Singh Sengar", 21);
28        try {
29            System.out.println("\nSerializing\n");
30            FileOutputStream file = new FileOutputStream("dataFile.txt");
31            ObjectOutputStream out = new ObjectOutputStream(file);
32            out.writeObject(d1);
33            out.close();
34            file.close();
35            System.out.println("Data before deserialization:\n ");
36            display(d1);
37        }
38        catch (Exception e) {
39            System.out.println("Exception Caught");
40        }
41        d1 = null;
42        try {
43            System.out.println("\nDeserializing\n");
44            FileInputStream file = new FileInputStream("dataFile.txt");
45            ObjectInputStream in = new ObjectInputStream(file);
46            d1 = (Data) in.readObject();
47            in.close();
48            file.close();
49            System.out.println("\n\nData after deserialization:\n ");
50            display(d1);
51        }
52        catch (Exception e) {
53            System.out.println("Exception Caught");
54        }
55    }
56 }
```

Serializing

ID Data: 1
Name Data: Vishnu Singh Sengar
Age Data: 21

Deserializing

Data after deserialization:

ID Data: 0
Name Data: Vishnu Singh Sengar
Age Data: 0

Figure 6 Code + Output for Question 4 (Hands On)

The serialized data file:

```
-í sr [ handsOnQuestions.DataÓ;í-!Â ] L namet Ljava/lang/String;L serialVersionUID Ljava/util/UUID;xpt
Vishnu Singh Sengarsr
java.util.UUID¼™]÷~m.../ J
leastSigBitsJ
mostSigBitsxp

d
```

End of Document