**MID TERM PAPER**

**Q.1**

**Assume our CSE department friend, Mr. JP, is interested in learning Java Programming. Assist him in understanding the output of the following queries within the jshell environment. Without explanation no marks will be awarded. Assume that Integer y= 10 and String s= "sggs" are predefined in the environment.**

**a. jshell> Integer.rotateLeft(y,3)**

**Ans:  Output:** 80

**Explanation:** int Integer.rotateLeft(int i, int distance) is the method in Integer class which is used to rotate 2’s complement binary representation of the specified int value left by specified number of bits.

i - the value whose bits are to be rotated left

distance - the number of bit positions to rotate left

Here y = 10 which is equal to 0001 0000 in binary and when it is rotated left by 3 bits it gives 1000 0000 which is equal to 80.

**b. jshell> s = (String) "sggs"**

**Ans: Output:** s= ‘sggs’

**Explanation:** This line assigns the string “sggs” to the variable s. The (String) casting is redundant here because “sggs” is already a String. The output s ==> sggs indicates that the value “sggs” has been assigned to the variable s

**c. jshell> y+(y=17)**

**Ans: Output:** 27

**Explanation:** This line demonstrates the behaviour of expressions and assignment in Java. The expression y + (y=17) first evaluates the right-hand side expression (y=17), which assigns the value 17 to the variable y. Then, it evaluates y (now 17) and adds it to the original value of y, which is 10. So, the result is 10 + 17 = 27

**d. jshell> y = Integer(25)**

**Ans:** **Error:** cannot find symbol

**Symbol:** method Integer(int)

**Explanation:** Integer(25) is not the correct syntax to create an Integer object. Instead, it should be y = new Integer(25). Because Integer is not a method it is a constructor hence we need to initialize it by using new operator.

**e. jshell> (short) 9042024**

**Ans: Output:** -1944

**Explanation:** The value 9042024 cannot be stored in a short variable because it is outside the range of a short datatype in Java.

Hence, when it is cast to short, it will overflow. In java, integer overflow wraps around, so the result will be -1944. Size of short is (2^16) that is 65536.

**In jshell –**

jshell> Math.pow(2,16)

$5 ==> 65536.0

jshell> 9042024 / 65536

$6 ==> 137

jshell> 9042024 % 65536

$7 ==> 63592

jshell> 63592 - 65536

$8 ==> -1944

OR

jshell> (short)9042024

$4 ==> -1944

**Q.2.**

**In what situations can the following errors occur1 in Java programming? Provide a brief example with short explanations.**

**a. error: Could not find or load main class Sample**

**Ans:** This error occurs when the JVM cannot locate the main class specified in the command line.

**Example –**

class A {

public static void main(String[] args) {

        System.out.println("Hello, world!");

}

}

In this code there is a class A and the file is saved as sample.java and if we provide a command in while compiling and running javac sample.java and java Sample but there is no Sample class hence it will give the error  “Could not find or load main class Sample”

**b. midterm.java:3: error: unclosed string literal**

**Ans:** This error occurs when there is a string literal in the code that is not closed properly with a closing quotation mark (")

public class midterm {

public static void main(String[] args) {

     String message = "Hello, world!;  // Missing closing quotation mark

        System.out.println(message);

}

}

**c. midterm.java:5: error: array required, but String found**

**Ans:** This error occurs when you attempt to access an element of an array, but the expression preceding the array index is not an array type.

public class midterm {

public static void main(String[] args) {

     String str = "Hello";

        System.out.println(str[0]); // Error: str is not an array, but used like one

}

}

In this example, str is a String variable, not an array. However, we're trying to access its first character using array notation (str[0]). This will result in the error "array required, but String found".

**Q.3.**

**How can we run the java code like C code? Provide the necessary commands for compiling and running the java code without implementing any class in a source file, assuming a JDK version 21 is available. Provide the meaning/purpose of commands in short.**

**Ans:**

To run Java code like C code without implementing any class in a source file, you can use a feature called "single-file source code execution" available in JDK 11 and later versions. This feature allows you to write Java code directly in a file without needing to define a class explicitly.

We can compile and run java code like c code using following commands -

compile ==> javac --release 21 --enable-preview fileName.java

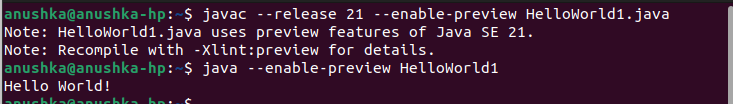
run ==> java --enable-preview filename

**Example -**

void main() {

System.out.println("Hello World!");

}

These commands allow you to compile and run Java code without needing to define a class explicitly, similar to how you would compile and run C code. The --enable-preview flag is used to enable any preview language features that may be used in the code.

**Q.4.**

**What will be the output of the following code? Analyse the code carefully. No partial marking will be provided. Short justification is required. Assume that the code is executed using java command and a class name in which main method is defined.**

**a. public class MidTerm {**

**MidTem() {**

**this(5);**

**}**

**MidTerm(int x) {**

**this();**

**}**

**public static void main(String[] args) {**

**MidTerm q3 = new MidTerm();**

**System.out.println("Welcome!");**

**}**

**}**

**Ans:** **Output:** Compilation Error

**Justification:** The constructor MidTerm() calls this(5) which in turn calls MidTerm(int x). However, MidTerm(int x) calls this() again, resulting in an infinite loop of constructor calls.Which is also called as constructor chaining. This causes a compilation error due to a recursive constructor invocation.

**b. public class Question {**

**public static void main(String[] args){**

**String exam = "Mid";**

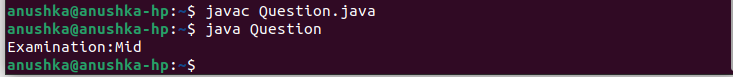
**exam.concat(" Semester");**

**System.out.println("Examination:" + exam);**

**}**

**}**

**Ans: Output:** Examination: Mid



**Justification:** The concat() method in Java String class does not modify the original string, but instead returns a new String with the concatenated value. However, the returned value is not assigned to any variable or printed, so the original value of exam remains unchanged, which is "Mid". Therefore, the output will be "Examination: Mid".

**c. public class Question{**

**Question4 g4=new Question4();**

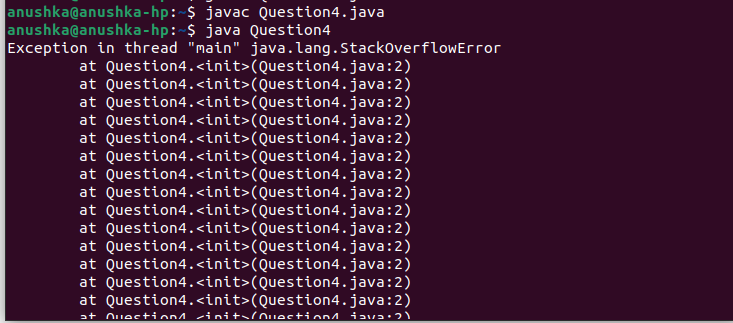
**public static void main(String[] args) {**

**Question4 q4=new Question4();**

**}**

**}**

**Ans: Output:** StackOverflowError



**Justification:** In the class Question, an instance of Question4 is created before its definition, resulting in a error due to the usage of Question4 before it is defined.

**d. class Exam{**

**static{**

**System.out.println("Static Block 1: EXAM");**

**}**

**Exam() {**

**System.out.println("EXAM Constructor");**

**}**

**{**

**System.out.println("Init Block: EXAM");**

**}**

**static {**

**System.out.println("Static Block 2: EXAM");**

**}**

**}**

**class MidTerm extends Exam{**

**static{**

**System.out.println("Static Block 1:MIDTERM");**

**}**

**MidTerm() {**

**System.out.println("MIDTERM Constructor");**

**}**

**public String toString() {**

**return "MidTerm";**

**}**

**static{**

**System.out.println("Static Block 2: MIDTERM");**

**}**

**public static void main(String args[]){**

**System.out.println("Welcome To Mid Term Examination");**

**Exam exam= new Exam();**

**MidTerm javaProgramming = new MidTerm();**

**System.out.println(javaProgramming);**

**}**

**{**

**System.out.println("Init Block: MIDTERM");**

**}**

**}**

**Ans: Output:** Static Block 1: EXAM

Static Block 2: EXAM

Static Block 1:MIDTERM

Static Block 2: MIDTERM

Welcome To Mid Term Examination

Init Block: EXAM

EXAM Constructor

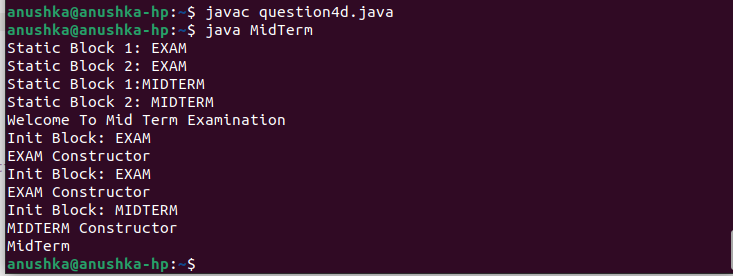
Init Block: EXAM

EXAM Constructor

Init Block: MIDTERM

MIDTERM Constructor

MidTerm



**Justification:** Static Blocks are executed when a class is loaded into memory.In Exam, Static Block 1: EXAM and Static Block 2: EXAM are printed first. Thern Static Block 1:MIDTERM and Static Block 2: MIDTERM are printed.

Then the main method is executed in which first print statement is Welcome To Mid Term Examination, it is printed.

Constructors are called when object is created. Hence when exam object is created it calls a constructor but Init Block is always called before calling a constructor. Init Block: EXAM and EXAM Constructor are executed.

Again when javaProgramming object is created it calls the MidTerm constructor but MidTerm class extends Exam class hence it again class Exam() constructor and before it Init block of exam. Then it calls MidTerm constructor and init block before calling constructor. Hence it prints following lines -  Init Block: EXAM

EXAM Constructor

Init Block: MIDTERM

MIDTERM Constructor

Then in System.out.println() it is printing the object itself but the class contains toString method hence it converts the address of object into the string which toString method is returning. Hence it prints MidTerm

**Q.5.**

**How many String objects are created at runtime? Justification is required.**

**public class Question{**

**public static void main(String[] args){**

**String str1= "Mid";**

**String str2= new String(“Term”);**

**String str3 = "Mid";**

**String str4 =str2;**

**String str5=new String(“Term”);**

**}**

**}**

**Ans:** 3 string objects will be created - str1, str3, str5

**Justification:**

str1 = "Mid";

The string literal "Mid" is used to initialize str1. String literals are stored in the string pool. So, a new String object will be created in the string pool.

str2 = new String("Term");

This line explicitly creates a new String object with the value "Term". It bypasses the string pool and creates a new object in the heap memory.

str3 = "Mid";:

This line also uses the string literal "Mid". A string object with String literal “Mid” is already created as str1, hence, it will refer to the str1. No new String object is created here.

str4 = str2;:

str4 is assigned the reference of str2, which is a String object created with new String("Term"). No new String object is created here.

str5 = new String("Term");:

This line explicitly creates another new String object with the value "Term". Similar to str2, it bypasses the string pool and creates a new object in the heap memory.