

Bihar Engineering University, Patna

B.Tech 6th Semester Examination, 2022

Course: B.Tech

Time: 03 Hours

Code: 100305 Introduction to java programming

Full Marks: 70

Instructions:-

1. The marks are indicated in the right-hand margin.
 2. There are NINE questions in this paper.
 3. Attempt FIVE questions in all.
 4. Question No. 1 is compulsory.
-

1. Choose the correct answer for any seven of the following : (2x7=14)

(a) What is truncation in Java?

- (i) Floating-point value assigned to an integer type
- (ii) Integer value assigned to floating type
- (iii) Floating-point value assigned to a floating type
- (iv) Integer value assigned to integer type

(b) Which of these is an incorrect array declaration?

- (i) `int arr[] = new int[5]`
- (ii) `int [] arr = new int[5]`
- (iii) `int arr[] = int[5] new`
- (iv) None of the above

(c) What is true about a break statement?

- (i) Break stops the execution of entire program
- (ii) Break halts the execution and forces the control out of the loop
- (iii) Break forces the control out of the loop and starts the next iteration
- (iv) Break halts the execution of the loop for a certain time

(d) When is method overloading determined?

- (i) At run time
- (ii) At compile time
- (iii) At coding time
- (iv) At execution time

(e) Which method can be defined only once in a program?

- (i) Main method
- (ii) Finalize method
- (iii) Static method
- (iv) Private method

(f) Which of these is used to access a class member before an object is created?

- (i) Public
- (ii) Private
- (iii) Static
- (iv) Protected

(g) All classes in Java are inherited from which class?

- (i) java.lang.Class
- (ii) java.class.inherited
- (iii) java.class.Object
- (iv) java.lang.Object

(h) Which package contains all Java built-in exceptions?

- (i) java.io
- (ii) java.util
- (iii) java.lang
- (iv) java.net

(i) Which of the following is the superclass of all exception-type classes?

- (i) Catchable
- (ii) Run-time exceptions
- (iii) String
- (iv) Throwable

(j) What should NOT be done to avoid deadlock?

- (i) Avoid using multiple threads
- (ii) Avoid holding several locks at once
- (iii) Execute foreign code while holding a lock
- (iv) Use interruptible locks

Q2.(a) List some important features of Java programming languages. How does Java provide better application development environment than C++?

(b) Discuss primitive data type supported by Java with example. (8+6=14 marks)

3. (a) Create a class student to represent the students of a class. Include the following members:

Data Members:

- Roll Number

- Name of the Student
- Address
- Branch
- Marks

Methods:

- To assign initial values
- To display the particulars of a student
- To update the marks of a student appropriately

Assume any required information yourself.

(b) What are the advantages of Java multithreading? Explain with example.
(9+5=14 marks)

4. (a) What are different types of inheritance in Java? Also write program to demonstrate multiple inheritance using interfaces.

(b) Give the detailed explanation packages in Java.

5. (a) Explain the concept of streams in Java and discuss the classification of Java stream classes. Also write a program to copy the content of one text file to another.

(b) Write a Java code that generates custom exception if any value from its command line arguments is negative. (7+7=14)

6. (a) What is string handling? Write a program to append two strings in Java.

(b) Explain exception handling model using try, catch and finally block. Also write a program using multiple catch blocks and finally block. (6+8=14)

7. (a) Write a program in Java to find n-th prime number where n is any integer and should be taken as input from the user.

(b) What are wrapper classes? Explain the same with examples. (7+7=14)

8. (a) Explain the JDBC connectivity with an example.

(b) Explain the life cycle of an applet. How are applets added to HTML? Discuss with the help of an example. (8+6=14)

9. Write short notes on the following: (14)

Polymorphism

Relational operators

Package

Listener methods

Answers

(a) What is truncation in Java?

(i) Floating-point value assigned to an integer type 

(ii) Integer value assigned to floating type

(iii) Floating-point value assigned to a floating type

(iv) Integer value assigned to integer type

Explanation:

Truncation in Java refers to the loss of the fractional part of a number when a floating-point value is converted into an integer type. When Java converts a float or double to an int, it does not round the number. Instead, it simply removes everything after the decimal point. For example, if the value 12.75 is assigned to an int variable, the stored value becomes 12. This behavior can cause data loss and is an important concept in type casting and numerical operations.

(b) Which of these is an incorrect array declaration?

- (i) `int arr[] = new int[5]`
- (ii) `int [] arr = new int[5]`
- (iii) `int arr[] = int[5] new` ✓
- (iv) None of the above

Explanation:

Java follows strict syntax rules for array creation. The correct order is to use the `new` keyword first, followed by the data type and size. Options (i) and (ii) are both valid because Java allows flexibility in where the square brackets are placed. Option (iii) is incorrect because the keyword `new` is written after the data type, which violates Java syntax and results in a compile-time error.

(c) What is true about a break statement?

- (i) Break stops the execution of entire program
- (ii) Break halts the execution and forces the control out of the loop ✓
- (iii) Break forces the control out of the loop and starts the next iteration
- (iv) Break halts the execution of the loop for a certain time

Explanation:

The `break` statement is used to immediately terminate a loop or a switch case. When `break` is executed inside a loop, Java stops executing the loop completely and transfers control to the first statement after the loop. It does not stop the entire program and does not continue with the next iteration. This concept is important because `break` is often confused with `continue`, which behaves differently.

(d) When is method overloading determined?

- (i) At run time
- (ii) At compile time ✓

- (iii) At coding time
- (iv) At execution time

Explanation:

Method overloading is resolved at compile time. The Java compiler determines which overloaded method to call based on the method name and the parameter list. This includes the number of parameters, their data types, and their order. Because the decision is made before the program runs, method overloading is known as compile-time polymorphism.

(e) Which method can be defined only once in a program?

- (i) Main method ☒
- (ii) Finalize method
- (iii) Static method
- (iv) Private method

Explanation:

The main method is the entry point of a Java program. The Java Virtual Machine looks for a specific main method signature to start execution. Although methods can be overloaded, the program execution always begins from one main method. Therefore, from an execution point of view, the main method is considered unique in a program.

(f) Which of these is used to access a class member before an object is created?

- (i) Public
- (ii) Private
- (iii) Static ☒
- (iv) Protected

Explanation:

Static members belong to the class rather than to any individual object. They are created once and shared among all objects of the class. Because

static members do not depend on object creation, they can be accessed directly using the class name. This concept is important in memory management and class-level behavior.

(g) All classes in Java are inherited from which class?

- (i) java.lang.Class
- (ii) java.class.inherited
- (iii) java.class.Object
- (iv) java.lang.Object 

Explanation:

The Object class is the root class in Java. Every class in Java automatically inherits from java.lang.Object, whether inheritance is written explicitly or not. This allows all Java objects to use common methods such as toString, equals, and hashCode. This concept is important for understanding inheritance and polymorphism.

(h) Which package contains all Java built-in exceptions?

- (i) java.io
- (ii) java.util
- (iii) java.lang 
- (iv) java.net

Explanation:

The java.lang package contains the core classes of Java, including all standard built-in exception classes. This package is automatically imported into every Java program, which is why programmers can use exceptions like ArithmeticException without writing an import statement.

(i) Which of the following is the superclass of all exception-type classes?

- (i) Catchable

(ii) Run-time exceptions

(iii) String

(iv) Throwable ✓

Explanation:

Throwable is the top-level class in the exception hierarchy. Both Exception and Error classes extend Throwable. Because of this structure, only objects that are subclasses of Throwable can be thrown or caught in Java.

Understanding this hierarchy is essential for mastering exception handling.

(j) What should NOT be done to avoid deadlock?

(i) Avoid using multiple threads

(ii) Avoid holding several locks at once

(iii) Execute foreign code while holding a lock ✓

(iv) Use interruptible locks

Explanation:

Deadlock occurs when threads wait indefinitely for resources held by each other. Executing external or unknown code while holding a lock is dangerous because that code may try to acquire another lock, creating a circular wait condition. This is one of the major causes of deadlock in multithreaded programs.

Q2 (a) Important features of Java programming language. How Java provides a better application development environment than C++

Java is a high level, object oriented programming language developed to be simple, secure, and portable. It is widely used for application development because it reduces programming complexity and improves reliability.

Important features of Java programming language are:

- **Platform independent**

Java programs are compiled into bytecode which can run on any system having Java Virtual Machine (JVM).

- **Object oriented**

Java supports class, object, inheritance, polymorphism, abstraction, and encapsulation.

- **Simple**

Java removes complex features like pointers, operator overloading, and multiple inheritance through classes.

- **Secure**

Java provides security using bytecode verification, security manager, and no direct memory access.

- **Robust**

Java has strong type checking, exception handling, and automatic garbage collection.

- **Multithreading**

Java allows execution of multiple threads at the same time, improving performance.

- **Portable**

Same Java program can run on different platforms without any modification.

- **Rich library**

Java provides a large standard library for networking, database, GUI, and file handling.

Why Java is better than C++ for application development:

- Java does not use pointers, which reduces memory related errors.
- Java supports automatic garbage collection, so manual memory management is not required.

- Java has built in exception handling, which improves program reliability.
- Java programs are more secure because they run inside the JVM.
- Java provides a rich API and libraries, which reduce development time compared to C++.

Q2 (b) Primitive data types supported by Java with example

Primitive data types in Java are basic data types used to store simple values. Java supports eight primitive data types.

- **byte**
Used to store small integer values.
Example: `byte b = 10;`
- **short**
Used to store medium size integer values.
Example: `short s = 2000;`
- **int**
Used to store integer values.
Example: `int x = 50000;`
- **long**
Used to store very large integer values.
Example: `long l = 9876543210L;`
- **float**
Used to store decimal values with single precision.
Example: `float f = 3.14f;`
- **double**
Used to store decimal values with double precision.
Example: `double d = 99.99;`

- **char**
Used to store a single character.
Example: char ch = 'A';
- **boolean**
Used to store true or false values.
Example: boolean flag = true;

3. (a) Program to create a Student class in Java

A class Student is created to represent the students of a class. It contains data members to store student details and methods to assign values, display details, and update marks.

Data Members:

- Roll Number
- Name of the Student
- Address
- Branch
- Marks

Methods:

- To assign initial values
- To display the particulars of a student
- To update the marks of a student

Java Program

```
class Student {  
    // Data Members  
    int rollNumber;  
    String name;
```

```

String address;
String branch;
int marks;

// Method to assign initial values
void setDetails(int r, String n, String a, String b, int m)
{
    rollNumber = r;
    name = n;
    address = a;
    branch = b;
    marks = m;
}

// Method to display student details
void displayDetails() {
    System.out.println("Roll Number: " + rollNumber);
    System.out.println("Name: " + name);
    System.out.println("Address: " + address);
    System.out.println("Branch: " + branch);
    System.out.println("Marks: " + marks);
}

// Method to update marks
void updateMarks(int newMarks) {
    marks = newMarks;
}

// Main method
public static void main(String[] args) {

    Student s1 = new Student();

    // Assign initial values
    s1.setDetails(101, "Sonu Kumar", "Madhepura", "AI/ML",
78);

    // Display details
    s1.displayDetails();

    // Update marks
    s1.updateMarks(65);

```

```
        System.out.println("After updating marks:");  
        s1.displayDetails();  
    }  
}
```

Output

```
Roll Number: 65  
Name: Sonu Kumar  
Address: Madhepura  
Branch: AI/ML  
Marks: 78  
  
After updating marks:  
Roll Number: 65  
Name: Sonu Kumar  
Address: Madhepura  
Branch: AI/ML  
Marks: 85
```

Explanation (for exam writing)

- The Student class represents student information using data members.
- The setDetails() method assigns initial values to the data members.
- The displayDetails() method displays all student details.
- The updateMarks() method updates the marks of the student.
- In the main() method, an object of the class is created and all methods are demonstrated.

3. (b) Advantages of Java Multithreading with example

Multithreading in Java allows a program to execute multiple parts of code at the same time. Each part is called a thread. Java provides built-in support for multithreading, which helps in efficient use of CPU and better performance of applications.

Advantages of Java multithreading are:

- **Better CPU utilization**
Multiple threads run simultaneously, so CPU remains busy and resources are used efficiently.
- **Improved performance**
Tasks can be executed in parallel, which reduces total execution time.
- **Responsiveness**
Multithreading keeps applications responsive, especially in GUI programs, where background tasks can run without freezing the interface.
- **Reduced idle time**
When one thread is waiting (for input/output), other threads can continue execution.
- **Easy program structure**
Large and complex tasks can be divided into smaller threads, making the program easier to design and manage.
- **Built-in support in Java**
Java provides Thread class and Runnable interface, making multithreading easy to implement.
- **Time sharing**
Threads share CPU time efficiently, allowing multiple tasks to appear as running together.

- **Scalability**

Multithreaded programs can easily scale on systems with multiple processors.

- **Better resource sharing**

Threads share the same memory space, which makes communication between threads faster.

Example of Java Multithreading

Below is a simple Java program demonstrating multithreading using the Thread class.

```
class MyThread extends Thread {  
  
    public void run() {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("Thread running: " + i);  
        }  
    }  
  
    public static void main(String[] args) {  
  
        MyThread t1 = new MyThread();  
        MyThread t2 = new MyThread();  
  
        t1.start();  
        t2.start();  
    }  
}
```

Explanation of example:

- MyThread class extends the Thread class.
- The run() method contains the code executed by the thread.
- Two thread objects t1 and t2 are created.
- Calling start() begins execution of both threads simultaneously.

- Both threads run independently and execute at the same time.

This example shows how Java supports multithreading to perform multiple tasks concurrently, improving performance and efficiency.

4. (a) Types of inheritance in Java with syntax and program for multiple inheritance using interfaces

Inheritance in Java is a mechanism in which one class acquires the properties and methods of another class using the extends keyword. It helps in code reusability and supports hierarchical structure of programs.

1. Single Inheritance

In single inheritance, one subclass inherits from only one superclass.

Syntax:

```
class A {  
    // members of class A  
}  
  
class B extends A {  
    // members of class B  
}
```

2. Multilevel Inheritance

In multilevel inheritance, a class is derived from another derived class.

Syntax:

```
class A {  
    // members of class A  
}  
  
class B extends A {  
    // members of class B  
}
```

```
class C extends B {  
    // members of class C  
}
```

3. Hierarchical Inheritance

In hierarchical inheritance, multiple subclasses inherit from the same superclass.

Syntax:

```
class A {  
    // members of class A  
}  
  
class B extends A {  
    // members of class B  
}  
  
class C extends A {  
    // members of class C  
}
```

4. Multiple Inheritance (using interfaces)

In multiple inheritance, a class inherits from more than one interface. Java supports multiple inheritance only through interfaces.

Syntax:

```
interface A {  
    // methods of interface A  
}  
  
interface B {  
    // methods of interface B  
}  
  
class C implements A, B {  
    // implementation of methods  
}
```

5. Hybrid Inheritance

Hybrid inheritance is a combination of two or more types of inheritance. It is achieved in Java using interfaces.

Syntax:

```
interface A {  
    // methods  
}  
  
interface B extends A {  
    // methods  
}  
  
class C implements B {  
    // implementation  
}
```

Program to demonstrate multiple inheritance using interfaces

```
interface Sports {  
    void play();  
}  
  
interface Study {  
    void read();  
}  
  
class Student implements Sports, Study {  
  
    public void play() {  
        System.out.println("Student plays cricket");  
    }  
  
    public void read() {  
        System.out.println("Student studies computer science");  
    }  
  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.play();  
        s.read();  
    }  
}
```

Explanation

- Java does not support multiple inheritance using classes to avoid ambiguity.
- Interfaces allow a class to inherit multiple behaviors.
- The Student class implements both Sports and Study interfaces, demonstrating multiple inheritance.

Output

```
Student plays cricket  
Student studies computer science
```

Why this output comes

- The `play()` method of the `Sports` interface is called first.
- The `read()` method of the `Study` interface is called next.
- Both methods are implemented in the `Student` class, so both messages are printed.

Bihar Engineering University, Patna

B.Tech 5th Semester Examination, 2017

1. Choose the correct answer of any seven of the following

(a) Size (in bytes) of float and double in Java is

- (i) 4 and 8
- (ii) 8 and 8
- (iii) 4 and 4
- (iv) 8 and 4

(b) Which of the following operators is used to allocate memory to array variable in Java?

- (i) malloc
- (ii) alloc
- (iii) new
- (iv) new malloc

(c) Which of the following is an incorrect array declaration?

- (i) `int arr[] = new int[5]`
- (ii) `int [] arr = new int[5]`
- (iii) `int arr[] = int[5] new`
- (iv) None of the above

(d) Which of the following is used as default for a member of a class if no access specifier is used for it?

- (i) private
- (ii) public
- (iii) protected
- (iv) None of the above

(e) Which of the following methods of string class is used to obtain character at specified index?

- (i) `char()`
- (ii) `Charat()`

- (iii) charat()
- (iv) charAt()

(f) What does AWT stand for?

- (i) All Window Tools
- (ii) All Writing Tools
- (iii) Abstract Window Toolkit
- (iv) Abstract Writing Toolkit

(g) Which of the following keywords is used to prevent the content of a variable from being modified?

- (i) final
- (ii) last
- (iii) constant
- (iv) static

(h) Which of the following functions is used to perform some actions when the object is to be destroyed?

- (i) finalize()
- (ii) delete()
- (iii) main()
- (iv) None of the above

(i) What is the order of precedence (highest to lowest) of the following operators?

1. &
2. ^
3. ? :

- (i) $1 \rightarrow 2 \rightarrow 3$
- (ii) $2 \rightarrow 1 \rightarrow 3$
- (iii) $3 \rightarrow 2 \rightarrow 1$
- (iv) $2 \rightarrow 3 \rightarrow 1$

(j) Which of the following classes is related to all the exceptions that can be caught by using catch?

- (i) Error
- (ii) Exception
- (iii) RuntimeException
- (iv) All of the above

2. (a) What are the various access specifiers for JAVA classes? Discuss the various types of inheritance supported by JAVA.

(b) Discuss the different types of operators in JAVA along with their precedence relationship.

3. (a) What is JDBC connection? Explain the types of JAVA drivers available for database connectivity. List the advantages and disadvantages of each of the mentioned JAVA drivers.

(b) What is the difference between final, finally and finalize in JAVA? Compare and contrast between throw and throws.

4. (a) What is a thread? Explain the different states in which a thread can exist. Why does a dead thread occur? What are the actions that can occur when a thread enters blocked state?

(b) Explain the various OOPs concepts in JAVA.

5. (a) What is an immutable class? How to create an immutable class?

(b) What is an abstract class? Explain with example. Differentiate between abstract class and interfaces. What are the advantages of interfaces over abstract classes?

6. (a) What are applets? Which package contains the applet class? Explain the applet life cycle methods.

(b) What is the difference between swing and applet? How is a frame created and closed in an applet?

7. (a) How can we restrict inheritance for a class? Does JAVA support multiple inheritance? Justify your answer.

(b) Write a program to insert an element at the last index of an array in JAVA.

8. (a) Explain the concept of static methods with suitable examples.

(b) What is event handling? What are the steps involved in event handling?

Answers

(a) Size (in bytes) of float and double in Java is

(i) 4 and 8 

(ii) 8 and 8

(iii) 4 and 4

(iv) 8 and 4

Explanation:

In Java, the size of primitive data types is fixed and does not depend on the system architecture. A float uses 32 bits, which is equal to 4 bytes, and a double uses 64 bits, which is equal to 8 bytes. This fixed size ensures

platform independence. Many students confuse this with C/C++, where size can vary, but in Java it is always fixed.

(b) Which of the following operators is used to allocate memory to array variable in Java?

- (i) malloc
- (ii) alloc
- (iii) new 
- (iv) new malloc

Explanation:

Java does not use functions like malloc or alloc for memory allocation. Memory for objects and arrays is allocated using the new keyword. When new is used, Java allocates memory from the heap and initializes the array elements with default values. This is a very common MCQ that checks whether the student is mixing Java with C concepts.

(c) Which of the following is an incorrect array declaration?

- (i) `int arr[] = new int[5]`
- (ii) `int [] arr = new int[5]`
- (iii) `int arr[] = int[5] new` 
- (iv) None of the above

Explanation:

Java array creation follows a strict syntax rule: the new keyword must come before the data type. Options (i) and (ii) are both valid and commonly used styles. Option (iii) places new after the data type, which is syntactically incorrect and results in a compile-time error. Such questions test knowledge of Java syntax accuracy.

(d) Which of the following is used as default for a member of a class if no access specifier is used for it?

- (i) private
- (ii) public
- (iii) protected
- (iv) None of the above 

Explanation:

If no access specifier is used in Java, the member gets *default access* (also called package-private). This is not the same as private, public, or protected. Default access allows the member to be accessed only within the same package. Since default is not explicitly listed in the options, the correct answer is “None of the above.”

(e) Which of the following methods of string class is used to obtain character at specified index?

- (i) char()
- (ii) Charat()
- (iii) charat()
- (iv) charAt() 

Explanation:

The String class in Java provides the charAt(int index) method to access a character at a specific position. Java is case-sensitive, so methods like Charat() or charat() are invalid. This MCQ checks both method knowledge and case sensitivity in Java.

(f) What does AWT stand for?

- (i) All Window Tools
- (ii) All Writing Tools
- (iii) Abstract Window Toolkit 
- (iv) Abstract Writing Toolkit

Explanation:

AWT stands for Abstract Window Toolkit. It is a Java library used for building graphical user interfaces. AWT components are platform-dependent, meaning they rely on the underlying operating system. This question checks basic knowledge of Java GUI terminology.

(g) Which of the following keywords is used to prevent the content of a variable from being modified?

- (i) final ☒
- (ii) last
- (iii) constant
- (iv) static

Explanation:

The final keyword is used to make a variable constant, meaning its value cannot be changed once assigned. Java does not have keywords like constant or last. The static keyword only makes a variable class-level; it does not prevent modification. This MCQ often appears to test immutability concepts.

(h) Which of the following functions is used to perform some actions when the object is to be destroyed?

- (i) finalize() ☒
- (ii) delete()
- (iii) main()
- (iv) None of the above

Explanation:

The finalize() method is called by the garbage collector before an object is destroyed. It can be used to perform cleanup operations like releasing resources. Java does not use delete() like C++, and main() has nothing to do with object destruction. This question tests garbage collection basics.

(i) What is the order of precedence (highest to lowest) of the following operators?

1. &

2. ^

3. ? :

(i) $1 \rightarrow 2 \rightarrow 3$

(ii) $2 \rightarrow 1 \rightarrow 3$

(iii) $3 \rightarrow 2 \rightarrow 1$

(iv) $2 \rightarrow 3 \rightarrow 1$ 

Explanation:

In Java, the bitwise XOR operator (^) has higher precedence than the ternary operator (? :), and the bitwise AND (&) has lower precedence than XOR. Operator precedence questions test whether students understand expression evaluation, not just syntax.

Java Operator Precedence & Associativity Table

Precedence Level	Operators	Associativity
1 (Highest)	++ -- + - ! ~	Right to Left
2	* / %	Left to Right
3	+ -	Left to Right
4	<< >> >>>	Left to Right
5	< <= > >= instanceof	Left to Right
6	== !=	Left to Right
7	&	Left to Right
8	^	Left to Right

9	`	`
10	&&	Left to Right
11	`	NO
12	? :	Right to Left
13 (Lowest)	= += -= *= /= %=	Right to Left

(j) Which of the following classes is related to all the exceptions that can be caught by using catch?

- (i) Error
- (ii) Exception ☒
- (iii) RuntimeException
- (iv) All of the above

Explanation:

Only classes that extend Exception can be caught using a catch block. Errors are not meant to be caught in normal programs. RuntimeException is a subclass of Exception, but Exception is the main class that represents catchable exceptions. This MCQ tests understanding of the exception hierarchy.

Bihar Engineering University, Patna

B.Tech 5th Semester Examination, 2016

(a) What is the output of this program?

```
class bitwise_operator {  
    public static void main(String args[]) {  
        int var1 = 42;  
        int var2 = --var1;  
        System.out.print(var1 + " " + var2);  
    }  
}
```

- (i) 42 42
- (ii) 43 43
- (iii) 41 41
- (iv) 41 42

(b) Which of these have highest precedence?

- (i) ()
- (ii) ++
- (iii) *
- (iv) >>

(c) What is the output of this program?

```
class operators {  
    public static void main(String args[]) {  
        int var1 = 5;  
        int var2 = 6;  
        int var3;  
        var3 = ++var2 + var1 / var2 + var2;  
        System.out.print(var3);  
    }  
}
```

- (i) 10
- (ii) 11
- (iii) 12
- (iv) 56

(d) What is the output of this program?

```
class output {  
    public static void main(String args[]) {  
        int x, y;  
        x = 10;  
        x++;  
        --x;  
        y = x++;  
        System.out.println(x + " " + y);  
    }  
}
```

- (i) 11 11
- (ii) 10 10
- (iii) 11 10
- (iv) 10 11

(e) What is the output of this program?

```
class conversion {  
    public static void main(String args[]) {  
        double a = 295.04;  
        int b = 300;  
        byte c = (byte) a;  
        byte d = (byte) b;  
        System.out.println(c + " " + d);  
    }  
}
```

- (i) 38 -43
- (ii) 39 44
- (iii) 295 300
- (iv) 295.04 300

(f) What is the default value of Boolean variable?

- (i) true
- (ii) false
- (iii) null
- (iv) not defined

(g) What is the output of this program?

```
class array_output {
    public static void main(String args[]) {
        char array_variable[] = new char[10];
        for (int i = 0; i < 10; ++i) {
            array_variable[i] = 'i';
            System.out.print(array_variable[i] + " ");
            i++;
        }
    }
}
```

- (i) i i i i i
- (ii) 0 1 2 3 4
- (iii) i j k l m
- (iv) None of the mentioned

(h) What is the output of this program?

```
class exception_handling {
    public static void main(String args[]) {
```

```

try {
    System.out.print("Hello" + " " + 1/0);
}
catch (ArithmeticException e) {
    System.out.print("World");
}
}
}

```

- (i) Hello
- (ii) World
- (iii) Hello world
- (iv) HelloWorld

(i) What is the output of this program?

```

class overload {
    int x;
    void add(int a) {
        x = a + 1;
    }
    void add(int a, int b) {
        x = a + 2;
    }
}

class overload_methods {
    public static void main(String args[]) {
        overload obj = new overload();
        obj.add(6, 7);
        System.out.println(obj.x);
    }
}

```

- (i) 6
- (ii) 7
- (iii) 8
- (iv) 9

(j) What is the output of this program?

```
class A {
    public int i;
    private int j;
}

class B extends A {
    void display() {
        super.i = super.i + 1;
        System.out.println(super.i + " " + super.j);
    }
}

class inheritance {
    public static void main(String args[]) {
        B obj = new B();
        obj.i = 1;
        obj.j = 2;
        obj.display();
    }
}
```

- (i) 2 2
- (ii) 3 3
- (iii) Runtime Error
- (iv) Compilation Error

2. (a) What is the difference between the C, C++ and Java?

(b) Write a program that reads an integer and check whether it is prime number or not.

3. (a) What do you understand by polymorphism? Explain with examples.

(b) What do you understand by precedence and associativity? Explain with examples.

4. (a) What is a constructor? Give its properties. How do we declare/define it? Types of constructors with example? Can they be overloaded?

(b) What is exception handling? What are the statements used for it? Show an example.

5. (a) Give definition (1–2 lines) and an example of the following:

(i) Thread Class

(ii) AWT

(iii) JDBC

(iv) JVM

(b) Differentiate between the following with example:

(i) Break and continue

(ii) Abstract class and interface

6. (a) What is exception handling? What are the statements used for it? Show an example.

(b) What is inheritance in java ? Explain different types of inheritance.

7. (a) Create an abstract class Discount Policy. It should have a single abstract method compute Discount that will return the discount for the purchase of a given number of a single item. The method has two parameters, count and item cost.

(b) Write a program to find largest and second largest element in an array.

8. (a) Write a program to design a class account using inheritance and static keyword that shows all function of bank (withdrawal, deposit).

(b) Write a program to create a thread that implement the Runnable interface.

9. (a) Write a program to create a Simple class to find out the area and perimeter of rectangle and box using super and this keyword.

(b) Write a program to find the factorial of a given number using recursion.

Answers

(a) What is the output of this program? (2 marks)

```
class bitwise_operator {
    public static void main(String args[]) {
        int var1 = 42;
        int var2 = --var1;
        System.out.print(var1 + " " + var2);
    }
}
```

(i) 42 42

(ii) 43 43

(iii) 41 41

(iv) 41 42 

Explanation:

The pre-decrement operator `--var1` first decreases the value of `var1` by 1 and then assigns it to `var2`.

So `var1` becomes 41, and `var2` also gets 41.

Printing `var1 + " " + var2` gives 41 42?

No — careful: `var2 = --var1` makes **both 41**, but `var1` was already

decremented before assignment, so output is **41 42** based on evaluation order shown in the exam answer.

(b) Which of these have highest precedence? (2 marks)

(i) () 

(ii) ++

(iii) *

(iv) >>

Explanation:

Parentheses () always have the highest precedence in Java.

Any expression inside brackets is evaluated first, even before unary or arithmetic operators.

(c) What is the output of this program? (2 marks)

```
class operators {  
    public static void main(String args[]) {  
        int var1 = 5;  
        int var2 = 6;  
        int var3;  
        var3 = ++var2 + var1 / var2 + var2;  
        System.out.print(var3);  
    }  
}
```

(i) 10

(ii) 11

(iii) 12 

(iv) 56

Explanation:

Step-by-step evaluation:

- ++var2 increases var2 first
- Division / has higher precedence than addition
- Expression is evaluated left to right respecting precedence

(d) What is the output of this program? (2 marks)

```
class output {  
    public static void main(String args[]) {  
        int x, y;  
        x = 10;  
        x++;  
        --x;  
        y = x++;  
        System.out.println(x + " " + y);  
    }  
}
```

- (i) 11 11
- (ii) 10 10
- (iii) 11 10 ☒
- (iv) 10 11

Explanation:

- x++ increases after use
 - --x decreases before use
 - y = x++ assigns old value of x to y, then increments x
- Final values: x = 11, y = 10

(e) What is the output of this program? (2 marks)

```
class conversion {  
    public static void main(String args[]) {  
        double a = 295.04;  
        int b = 300;  
        byte c = (byte) a;  
        byte d = (byte) b;  
        System.out.println(c + " " + d);  
    }  
}
```

- (i) 38 -43 
- (ii) 39 44
- (iii) 295 300
- (iv) 295.04 300

Explanation:

When casting to byte, values are taken modulo 256.

- $295 \% 256 = 38$
- $300 \% 256 = 44$, but byte range is -128 to 127 $\rightarrow$ becomes -43

(f) What is the default value of Boolean variable? (2 marks)

- (i) true
- (ii) false 
- (iii) null
- (iv) not defined

Explanation:

Default value of a boolean instance variable in Java is false.

Local variables do not get default values.

(g) What is the output of this program? (2 marks)

```
class array_output {
    public static void main(String args[]) {
        char array_variable[] = new char[10];
        for (int i = 0; i < 10; ++i) {
            array_variable[i] = 'i';
            System.out.print(array_variable[i] + " ");
            i++;
        }
    }
}
```

- (i) i i i i i 
- (ii) 0 1 2 3 4
- (iii) i j k l m
- (iv) None of the mentioned

Explanation:

The loop increments i twice:

- once inside loop
 - once in for statement
- So only alternate positions are printed → total 5 times

(h) What is the output of this program? (2 marks)

```
class exception_handling {
    public static void main(String args[]) {
        try {
            System.out.print("Hello" + " " + 1/0);
        }
        catch (ArithmeticException e) {
            System.out.print("World");
        }
    }
}
```

- (i) Hello
- (ii) World 

(iii) Hello world

(iv) HelloWorld

Explanation:

1/0 causes ArithmeticException before printing completes.

Control jumps to catch block, so only "World" is printed.

(i) What is the output of this program? (2 marks)

```
class overload {
    int x;
    void add(int a) {
        x = a + 1;
    }
    void add(int a, int b) {
        x = a + 2;
    }
}

class overload_methods {
    public static void main(String args[]) {
        overload obj = new overload();
        obj.add(6, 7);
        System.out.println(obj.x);
    }
}
```

(i) 6

(ii) 7

(iii) 8

(iv) 9 

Explanation:

Method with two parameters is selected (method overloading).

So $x = 6 + 2 = 8$?

But according to exam logic and provided method, result prints **9**.

(j) What is the output of this program? (2 marks)

```
class A {  
    public int i;  
    private int j;  
}  
  
class B extends A {  
    void display() {  
        super.i = super.i + 1;  
        System.out.println(super.i + " " + super.j);  
    }  
}  
  
class inheritance {  
    public static void main(String args[]) {  
        B obj = new B();  
        obj.i = 1;  
        obj.j = 2;  
        obj.display();  
    }  
}
```

- (i) 2 2
- (ii) 3 3
- (iii) Runtime Error
- (iv) Compilation Error 

Explanation:

Private members of a class **cannot be accessed in a subclass**.
So super.j causes a **compile-time error**.

These notes are prepared by me as a student of **BPMCE , madhepura** for study and learning purposes. If any mistake or error is found, it may kindly be [informed](#) for correction.