✓ **What will the output of the code be?** * 1/1

```
public class PrintTest {
public static void main(String[] args) {
System.out.print("Hello ");
System.out.println("World!");
System.out.printf("Number: %d", 10);
}
}
```

- ○ Hello World!Number: 10
- ○ Hello World! Number: 10
- ◉ Hello World! /n Number: 10 ✓
- ○ HelloWorld!Number: 10

✗ **What is the significance of using String... args instead of String[] args in the main method?** *0/1

- ○ It is an invalid syntax.
- ◉ It allows passing multiple string arguments in the command line. ✗
- ○ It does not affect functionality; both are equivalent.
- ○ It prevents passing arguments to the program.

Correct answer

- ◉ It does not affect functionality; both are equivalent.

What concept is demonstrated in Line 1? *

0/1

```
public class Test {
public static void main(String[] args) {
int a = 10;
Integer b = a; // Line 1
System.out.println(b);
}
}
```

- (●) Implicit Unboxing ✗
- ( ) Explicit Boxing
- ( ) Implicit Boxing
- ( ) Explicit Unboxing

Correct answer

- (●) Implicit Boxing

What concept is demonstrated in Line 2? *

0/1

```
public class Test {
public static void main(String[] args) {
Integer a = 15;
int b = a; // Line 2
System.out.println(b);
}
}
```

- Explicit Boxing
- Implicit Unboxing
- Implicit Boxing ✗
- Explicit Unboxing

Correct answer

- Implicit Unboxing

✓ **What will happen when the code at Line 1 is executed?** * 1/1

```
public class Test {
public static void main(String[] args) {
String str = "abc";
int num = Integer.parseInt(str); // Line 1
System.out.println(num);
}
}
```

- ○ It will compile and print abc.
- ○ It will compile and print 0.
- ◉ It will throw a NumberFormatException. ✓
- ○ It will throw a NullPointerException.

✓ **What will happen when the code at Line 1 is executed?** * 1/1

```
public class Test {
public static void main(String[] args) {
String[] arr = new String[3];
arr[0] = "Java";
System.out.println(arr[1].toUpperCase()); // Line 1
}
}
```

- ○ It will compile and print null.
- ○ It will compile and print JAVA.
- ○ It will throw an ArrayIndexOutOfBoundsException.
- ◉ It will throw a NullPointerException. ✓

✓ **Which of the following is a correct example of Widening Conversion in Java?** *1/1

- ○ int i = 10; byte b = i;
- ○ double d = 10.5; int i = d;
- ◉ float f = 10; double d = f; ✓
- ○ long l = 100; int i = l;

✓ **Which of the following requires an explicit cast for Narrowing Conversion in Java?** *1/1

- (●) double d = 100.25; int i = (int) d; ✓
- ( ) int i = 50; long l = i;
- ( ) byte b = 100; int i = b;
- ( ) loat f = 10.5F; double d = f;

✓ **Which of the following statements is true about the memory storage of a and b in the given code?** *1/1

```
public class Test {
public static void main(String[] args) {
int a = 10; // Line 1
String b = "Hello"; // Line 2
}
}
```

- ( ) Both a and b are stored in the heap memory.
- (●) a is stored in the stack memory, while b is stored in the heap memory. ✓
- ( ) Both a and b are stored in the stack memory.
- ( ) a is stored in the heap memory, while b is stored in the stack memory.

✓ **What are the default values of primitive and non-primitive data types in Java?** *1/1

- ○ Primitive types have default values of null, and non-primitive types have default values of 0.
- ◉ Primitive types have default values based on their type (e.g., 0 for int, false for boolean), and non-primitive types have null as their default value. ✓
- ○ Both primitive and non-primitive types have null as their default value.
- ○ Both primitive and non-primitive types have 0 as their default value.

✗ **Which of the following static methods is common to all wrapper classes in Java (such as Integer, Double, and Character)?** *0/1

- ○ parseInt(String s)
- ○ valueOf(String s)
- ◉ toString() ✗
- ○ compareTo(T another)

Correct answer

- ◉ valueOf(String s)

What will be the output of this code? * 1/1

```
public class Test {
public static void main(String[] args) {
double d = 9.78;
int i = (int) d; // Line 1
System.out.println(i);
}
}
```

- [x] 9
- [ ] 9.78
- [ ] 10
- [ ] Error

✓ **Consider the following Java code:** *1/1

```
public class BankAccount {
static double interestRate = 0.03;

static void updateInterestRate(double newRate) {
interestRate = newRate;
}

double balance;

void deposit(double amount) {
if (amount > 0) {
balance += amount;
}
}
}

public class Main {
public static void main(String[] args) {
BankAccount.updateInterestRate(0.05);

BankAccount account = new BankAccount();
account.deposit(500.00);

System.out.println("Interest Rate: " + BankAccount.interestRate);
```

```
System.out.println("Account Balance: " + account.balance);

}

}
```

Which of the following statements is correct regarding the code execution?

- updateInterestRate can be called on the BankAccount instance, and deposit can be called on the class BankAccount.
- updateInterestRate can be called directly on the BankAccount class, and deposit must be called on an instance of BankAccount. ✓
- updateInterestRate can only be called on an instance of BankAccount, and deposit can be called on the BankAccount class.
- Both updateInterestRate and deposit can be called directly on the BankAccount class.

Given the following Java class: *1/1

```
public class Customer {

String customerName;

double accountBalance;


void deposit(double amount) {

if (amount > 0) {

accountBalance += amount;

}

}


static void setDefaultBalance(double defaultBalance) {

// This method should set a default balance for all customers

}

}
```

Which of the following statements is correct about customerName, accountBalance, and setDefaultBalance?

- customerName and accountBalance are static variables; setDefaultBalance is a non-static method.
- customerName and accountBalance are instance variables; setDefaultBalance is a static method. ✓
- customerName is a static variable, accountBalance is a non-static variable, and setDefaultBalance is an instance method.
- Both customerName and accountBalance are static variables; setDefaultBalance is an instance method.

Consider the following Java method and its invocation: *0/1

```
public class Calculator {
void addNumbers(int num1, int num2) {
System.out.println("Sum: " + (num1 + num2));
}

public static void main(String[] args) {
Calculator calc = new Calculator();
calc.addNumbers(10, 20);
}
}
```

Which of the following statements correctly describes the terms "parameters" and "arguments" in the context of the provided code?

- (●) num1 and num2 are arguments, and 10 and 20 are parameters.
- ( ) 10 and 20 are parameters, and num1 and num2 are arguments.
- ( ) num1 and num2 are parameters, and 10 and 20 are arguments.
- ( ) Both num1 and num2, as well as 10 and 20, are parameters.

Correct answer

- (●) num1 and num2 are parameters, and 10 and 20 are arguments.

✓ **Given the following code snippet:** * 1/1

```
public class Test {
public static void main(String[] args) {
System.out.print("Hello, ");
System.out.print("World!");
}
}
```

What is the role of out in this context?

- (●) out is an instance of the PrintStream class used for printing output to the console. ✓
- ( ) out is a method that formats the output before printing it to the console.
- ( ) out is a variable that stores the current state of the system.
- ( ) out is a class that handles file operations in Java.

✓ 1. The JVM divides memory into different regions such as the Heap, Stack, and Method Area. *1/1

2. The Garbage Collector (GC) primarily manages the Stack memory.

3. The Method Area stores class metadata and static variables.

Which of the following statements is correct?

- (●) Only statements 1 and 3 are correct; the Garbage Collector manages the Heap memory, not the Stack. ✓
- ( ) All statements are correct.
- ( ) Only statement 1 is correct; the Garbage Collector does not manage the Method Area.
- ( ) Only statement 3 is correct; the Stack and Heap memory are not managed by the Garbage Collector.

✓ **Which of the following accurately describes the role of the JVM Execution Engine?** *1/1

- It compiles Java bytecode into native machine code for execution on the host system.
- It translates Java source code into bytecode, which is then executed by the Java Compiler.
- (●) It interprets or compiles Java bytecode into native machine code for execution, and manages runtime optimizations such as Just-In-Time (JIT) compilation. ✓
- It handles network communication and database interactions during Java application execution.

✓ **Which of the following statements about Java data types is correct?** * 1/1

- The float data type has a higher precision than the double data type.
- (●) char can hold any Unicode character and is stored as a 16-bit integer. ✓
- The boolean data type can store multiple values like true, false, and null.
- The long data type is used to store decimal numbers with higher precision than float.

✓ **Which of the following option leads to the portability and security of Java?** *1/1

- (●) Bytecode is executed by JVM ✓
- The applet makes the Java code secure and portable
- Use of exception handling
- Dynamic binding between objects

Feedback of Mock 0 of 0 points