OOPJ CCEE Practice Quiz - 2

Total points 26/40

Questions: 40
Time: 60 Mins
Extra Section: 5 Mins

Must analyse the concepts afterwards, don't procrastinate. Analysis & Learning = Success.

The respondent's email (**harshal.tarmale.cmaug25@gmail.com**) was recorded on submission of this form.

0 of 0 points

Name *

Harshal Vilas Tarmale

MCQs

26 of 40 points

✓ public class Test { * 1/1

```
    static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

- ○ Compiles and prints "Hello World"
- ◉ Compilation error ✓
- ○ Runtime error
- ○ Nothing happens

✓ 1/1

```
public class Test {
    public static void main(String... args) {
        System.out.println("Arguments: " + args.length);
    }
}
```

What is true about this main method declaration? *

- ○ Invalid syntax
- ◉ Valid, equivalent to String[] args ✓
- ○ Valid, but can only accept multiple arguments
- ○ Valid, but performance is slower

✗ public class Counter { * 0/1

```
    static int count = 0;
    int instanceCount = 0;

    public Counter() {
        count++;
        instanceCount++;
    }
    public static void main(String[] args) {
        Counter c1 = new Counter();
        Counter c2 = new Counter();
        System.out.println(count + " " + c1.instanceCount);
    }
}
```

What will be the output?

- ○ 1 1
- ◉ 2 2 ✗
- ○ 2 1
- ○ 1 2

Correct answer

- ◉ 2 1

✓ public class Golmaal { * 1/1

```
    static int x = 10;
    static void method() {
        int x = 20;
        System.out.print(x);
    }
    public static void main(String[] args) {
        method();
        System.out.print(x);
    }
}
```

What will be the output?

- ○ 1010
- ○ 2020
- ◉ 2010 ✓
- ○ Compilation error

✓ public class A { * 1/1

```
    static {
        System.out.print("A");
    }
    {
        System.out.print("B");
    }
    public A() {
        System.out.print("C");
    }
    static {
        System.out.print("D");
    }
    public static void main(String[] args) {
        new A();
        new A();
    }
}
```

What will be the output?

- (●) ADBCBC ✓
- ( ) BCBC
- ( ) ADBCAD
- ( ) ADBC

✓ public class Student { * 1/1

```
    String name;
    int age;
    public Student() {
        this("Unknown", 0);
    }
    public Student(String name) {
        this(name, 18);
    }
    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public static void main(String[] args) {
        Student s = new Student("Shaktimaan");
        System.out.println(s.name + " " + s.age);
    }
}
```

What will be the output?

- ○ Unknown 0
- ○ Shaktimaan 0
- ◉ Shaktimaan18 ✓
- ○ Compilation error

✓ public class Superman { * 1/1

```
  public Superman(int x) {
    System.out.println("Constructor: " + x);
  }
  public static void main(String[] args) {
    Superman s = new Superman();
  }
}
```

What will happen?

- ○ Prints "Constructor: 0"
- ◉ Compilation error ✓
- ○ Runtime error
- ○ Prints "Constructor: " + null

✓ public class HakunaMatata { * 1/1

```
    int value = 10;
    void setValue(int value) {
        value = value;
    }
    void setValueCorrect(int value) {
        this.value = value;
    }
    public static void main(String[] args) {
        HakunaMatata obj = new HakunaMatata();
        obj.setValue(20);
        System.out.print(obj.value + " ");
        obj.setValueCorrect(30);
        System.out.print(obj.value);
    }
}
```

What will be the output?

- (●) 10 30 ✓
- ( ) 20 30
- ( ) 10 10
- ( ) 20 20

✗ public class AreyArey { * 0/1

```
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3};
        int[] arr2 = arr1;
        arr2[0] = 10;
        System.out.println(arr1[0] + " " + arr2[0]);
    }
}
```

What will be the output?

- (●) 1 10 ✗
- ( ) 10 1
- ( ) 10 10
- ( ) 1 1

Correct answer

- (●) 10 10

✓ public class BadaArray { * 1/1

```
    public static void main(String[] args) {
        int[] arr = new int[5];
        System.out.println(arr[0] + " " + arr.length);
    }
}
```

What will be the output?

- ○ null 5
- ◉ 0 5 ✓
- ○ undefined 5
- ○ Compilation error

✓ public class JaneKahanGayeWohDin { * 1/1

```
    public static void main(String[] args) {
        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 2; j++) {
                if(i == j) continue;
                System.out.print(i + "" + j + " ");
            }
        }
    }
}
```

What will be the output?

- 01 10 12 21
- Compilation error
- 00 01 10 11 20 21 22
- 01 10 20 21 ✓

✓ public class YeDilDeewana { * 1/1

```
public static void main(String[] args) {
    int i = 1;
    while(i <= 3) {
        System.out.print(i + " ");
        i++;
    }
    System.out.print(i);
  }
}
```

What will be the output?

- ○ 1 2 3
- ◉ 1 2 3 4 ✓
- ○ 1 2 3 3
- ○ Infinite loop

✗ public class SwitchOnOff { * 0/1

```
public static void main(String[] args) {
    int x = 2;
    switch(x) {
        case 1:
            System.out.print("One ");
        case 2:
            System.out.print("Two ");
        case 3:
            System.out.print("Three ");
        default:
            System.out.print("Default");
    }
  }
}
```

What will be the output?

- (●) Two ✗
- ( ) Two Three Default
- ( ) Two Three
- ( ) Default

Correct answer

- (●) Two Three Default

✓ public class Potholes { * 1/1

```
public static void main(String[] args) {
    for(int i = 1; i <= 5; i++) {
        if(i == 2) continue;
        if(i == 4) break;
        System.out.print(i + " ");
    }
    System.out.print("Done");
}
}
```

What will be the output?

- (●) 1 3 Done ✓
- ( ) 1 2 3 Done
- ( ) 1 3 4 Done
- ( ) 1 3 5 Done

✓ public class Constructor { * 1/1

```
public static void main(String[] args) {
    String s1 = "Hello";
    String s2 = "Hello";
    String s3 = new String("Hello");
    System.out.print(s1 == s2);
    System.out.print(s1 == s3);
    System.out.print(s1.equals(s3));
  }
}
```

What will be the output?

- (●) true false true ✓
- ( ) false false true
- ( ) true true true
- ( ) false true true

✗ public class Assignment { * 0/1

```
public static void main(String[] args) {
    String s1 = "Yeh Dil Mange More";
    s1.concat(" Programming");
    System.out.println(s1);
  }
}
```

What will be the output?

- (●) Yeh Dil Mange More Programming ✗
- ( ) Yeh Dil Mange More
- ( ) Programming
- ( ) Compilation error

Correct answer

- (●) Yeh Dil Mange More

✗ public class MemoryLoss { * 0/1

```
public static void main(String[] args) {
    String s1 = "test";
    String s2 = new String("test");
    String s3 = s2.intern();
   System.out.println(s1 == s3);
  }
}
```

What will be the output and why?

- ○ false - s1 in heap, s3 in stack
- ○ true - both reference string pool
- ○ false - different memory locations
- ◉ Compilation error ✗

Correct answer

- ◉ true - both reference string pool

✓ 1/1

```
public class PhirMilengy {
   int count = 0;
 PhirMilengy() {
      count++;
      System.out.print(count + " ");
   }
 void display() {
      count++;
      System.out.print(count + " ");
   }
public static void main(String[] args) {
      PhirMilengy obj1 =  PhirMilengy();
      obj1.display();
     PhirMilengy obj2 = new PhirMilengy();
      obj2.display();
   }
}
```

What will be the output? *

- ○ 1 2 1 2
- ○ 1 2 3 4
- ○ 1 1 1 1
- ◉ Compilation error ✓

0/1

```
public class Promise {
   int x;
Promise() {
      this(10);
      System.out.print("A ");
   }
 Promise(int x) {
      this.x = x;
      System.out.print("B ");
   }
public static void main(String[] args) {
      Promise obj = new Promise();
      System.out.print(obj.x);
   }
}
```

What will be the output?

- (●) A B 10 ✗
- ( ) B A 10
- ( ) A B 0
- ( ) B A 0

Correct answer

- (●) B A 10

✗ public class BiggerGoal { * 0/1

```
    public static void main(String[] args) {
        int[] original = {1, 2, 3, 4, 5};
        int[] copy = original;
        int[] clone = original.clone();
 original[0] = 100;
 System.out.println(copy[0] + " " + clone[0]);
    }
}
```

What will be the output?

- (●) 1 1 ✗
- ( ) 100 100
- ( ) 100 1
- ( ) 1 100

Correct answer

- (●) 100 1

✓ public class Explore { * 1/1

```
    public static void main(String[] args) {
        outer: for(int i = 1; i <= 3; i++) {
            for(int j = 1; j <= 3; j++) {
                if(i == 2 && j == 2) continue outer;
                System.out.print(i + "" + j + " ");
            }
        }
    }
}
```

What will be the output?

- ○ Runtime error
- ○ 11 12 13 21 22 23 31 32 33
- ◉ 11 12 13 21 31 32 33 ✓
- ○ 11 12 21 31 32 33

✗ public class SwimmingPool { * 0/1

```
    public static void main(String[] args) {
        String s1 = "hello";
        String s2 = "hel" + "lo";
        String s3 = "hel";
        String s4 = s3 + "lo";
 System.out.println(s1 == s2);
        System.out.println(s1 == s4);
    }
}
```

What will be the output?

- (●) true true ✗
- ( ) false false
- ( ) true false
- ( ) false true

Correct answer

- (●) true false

✓ public class MainHoonNa { * 1/1

```
  static void method1() {
      System.out.print("Static ");
      method2();
  }
 void method2() {
      System.out.print("Instance");
  }
 public static void main(String[] args) {
      StaticCall.method1();
  }
}
```

What will happen?

- ○ Prints "Static Instance"
- ○ Prints "Static"
- ◉ Compilation error ✓
- ○ Runtime error

✓ public class Challenge { * 1/1

```
    static String msg = "Start";
    String instance = "Instance";
static {
        msg += " Static";
    }
 {
        instance += " Block";
    }
  Challenge() {
        instance += " Constructor";
    }
 public static void main(String[] args) {
        System.out.print(msg + " ");
        Challenge obj = new Challenge();
        System.out.print(obj.instance);
    }
}
```

What will be the output?

- (●) Start Static Instance Block Constructor ✓
- ( ) Start Instance Block Constructor
- ( ) Start Static Instance Constructor
- ( ) Static Instance Block Constructor

✗ * 0/1

```
public class StringEkPremKatha {
    public static void main(String[] args) {
        String s1 = new String("hello").intern();
        String s2 = "hello";
        String s3 = new String("hello");
      System.out.println((s1 == s2) + " " + (s2 == s3) + " " + s1.equals(s3));
    }
}
```

What will be the output?

- ○ true false true
- ◉ false false true ✗
- ○ true true true
- ○ false true false

Correct answer

- ◉ true false true

✕ *0/1

```
public class UltaPulta {
    static int staticVar = 0;
    int instanceVar = 0;
  {
        instanceVar++;
        staticVar++;
    }
  UltaPulta() {
        instanceVar++;
        staticVar++;
    }
   UltaPulta(int x) {
        this();
        instanceVar += x;
    }
    public static void main(String[] args) {
        UltaPulta obj1 = new UltaPulta();
        UltaPulta obj2 = new UltaPulta(5);
        System.out.println(staticVar + " " + obj1.instanceVar + " " +
obj2.instanceVar);
    }
}
```

What will be the output?

- ○ 4 2 7
- ◉ 4 2 8 ✕
- ○

6 2 8

4 1 6

Correct answer

4 2 7

✗ import java.time.LocalDate; * 0/1

```
import java.time.format.DateTimeFormatter;


public class DateTest {
   public static void main(String[] args) {
      LocalDate date1 = LocalDate.of(2024, 1, 15);
      LocalDate date2 = LocalDate.parse("2024-01-15");
  System.out.println(date1.equals(date2));
      System.out.println(date1 == date2);
   }
}
```

What will be the output?

true true

false false ✗

true false

false true

Correct answer

true false

✗ *0/1

```
import java.time.LocalDateTime;
import java.time.LocalDate;
import java.time.LocalTime;

public class DateNow {
   public static void main(String[] args) {
      LocalDate today = LocalDate.now();
      LocalTime currentTime = LocalTime.now();
      LocalDateTime now = LocalDateTime.now();
 LocalDateTime combined = LocalDateTime.of(today, currentTime);
      System.out.println(now.toLocalDate().equals(combined.toLocalDate(
)));
      System.out.println(now.equals(combined));
   }
}
```

What will be the output most likely be?

- (●) true true ✗
- ( ) false false
- ( ) true false
- ( ) false true

Correct answer

- (●) true false

✓ Why does the String literal pool exist in Java, and what is the primary benefit it provides? *1/1

- ○ To make string concatenation faster by avoiding object creation
- ◉ To reduce memory consumption by sharing identical string literals across the application ✓
- ○ To prevent strings from being garbage collected during program execution
- ○ To store strings in stack memory instead of heap memory for faster access

✓ What is the fundamental difference between how primitive variables and object references are stored in memory during method execution? *1/1

- ○ Primitives are stored in heap, objects in stack
- ◉ Primitives are stored in stack, object data in heap, object references in stack ✓
- ○ Both primitives and objects are stored in heap memory
- ○ Primitives are stored in method area, objects in heap

✓ In what order are static elements initialized when a class is first loaded by the JVM? *1/1

- ○ Static variables, static blocks, static methods (in declaration order)
- ○ Static blocks first, then static variables (in declaration order)
- ◉ Static variables and static blocks together in the order they appear in the class ✓
- ○ All static elements are initialized simultaneously by the JVM

✓ Why can't static methods access non-static (instance) variables directly? * 1/1

- ○ Static methods execute before objects are created
- ◉ Static methods belong to the class, not to any specific instance, so there's no 'this' context ✓
- ○ Static methods are stored in method area, instance variables in heap
- ○ It would cause memory leaks if static methods could access instance variables

✓ What is the primary purpose of constructor chaining using this() calls? * 1/1

- ○ To improve performance by reducing code duplication
- ◉ To ensure consistent object initialization and reduce code redundancy ✓
- ○ To allow multiple inheritance in Java classes
- ○ To enable polymorphism during object creation

✗ Why must this() call be the first statement in a constructor? * 0/1

- ○ To ensure proper memory allocation for the object
- ○ To prevent infinite recursion in constructor calls
- ○ To guarantee the object is fully initialized before any other operations
- ◉ To comply with Java syntax rules for method chaining ✗

Correct answer

- ◉ To guarantee the object is fully initialized before any other operations

✓ When you declare int[ ][ ] matrix = new int[3][4], what exactly gets created in memory? *1/1

- ○ A single contiguous block of 12 integers in heap memory
- ◉ One array object containing references to 3 separate array objects, each containing 4 integers ✓
- ○ 4 array objects in heap, each containing 3 integers
- ○ A two-dimensional structure stored directly in stack memory

✓ What is the fundamental difference between break and continue in terms of loop control flow? *1/1

- ○ break exits the current iteration, continue exits the entire loop
- ◉ break terminates the loop completely, continue skips to the next iteration ✓
- ○ break works with all loops, continue only works with for loops
- ○ break affects outer loops, continue affects inner loops

✓ Why did Java designers make String objects immutable, and what is the most significant benefit? *1/1

- ○ To improve string concatenation performance
- ◉ To enable string objects to be safely shared across multiple threads without synchronization ✓
- ○ To reduce memory usage by preventing string modifications
- ○ To make string comparison operations faster using == operator

✗ What is the complete sequence of events when new ClassName() is executed *0/1

- ○ Memory allocation → Constructor execution → Object reference return
- ○ Constructor execution → Memory allocation → Instance variable initialization
- ○ Memory allocation → Instance variable initialization → Instance blocks → Constructor
- ◉ Class loading → Memory allocation → Static blocks → Constructor ✗

Correct answer

- ◉ Memory allocation → Instance variable initialization → Instance blocks → Constructor

✓ How does the JVM determine which method to call when you have method overloading? *1/1

- ○ At runtime based on the actual object type (dynamic binding)
- ◉ At compile time based on the method signature and argument types (static binding) ✓
- ○ By checking the return type of the method being called
- ○ By using the most recently defined method with matching name

✓ Where are the following stored in JVM memory: static variables, instance variables, local variables, and method bytecode? *1/1

- ○ All in heap memory for easy garbage collection
- ◉ Static variables in method area, instance variables in heap, local variables in stack, method bytecode in method area ✓
- ○ Static variables in stack, instance variables in heap, local variables in stack, method bytecode in heap
- ○ All variables in stack memory, method bytecode in method area

Experience Section 0 of 0 points

Tough tasks create tough PERSONALITY? *

- ◉ Agree
- ○ Not Agree

Level of Exam *

- ○ Easy
- ◉ Medium
- ○ Hard

How is your experience? (No one word) *

Very intensive. Confuses a lot. need to improve string, static and date and time.

Google Forms