OOPJ CCEE Practice Quiz - 3

Total points 37/40

Total: 40 Questions
Duration: 1 hour
Experience Zone: 5 mins Extra

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

0 of 0 points

Name *

Harshal Vilas Tarmale

PRN *

250840320073

MCQ 37 of 40 points

✓ public class JugaadManager { * 1/1

```
public class JugaadManager   {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.print("ArithmeticException ");
        } catch (Exception e) {
            System.out.print("Exception ");
        } finally {
            System.out.print("Finally ");
        }
        System.out.print("End");
    }
}
```

What will be the output?

- [✓ ArithmeticException Finally End] ✓
- Exception Finally End
- ArithmeticException End
- Finally End

✓ 1/1

```
public class  ThodaAdjustKaro  {           *
    public static void main(String[] args) {
        System.out.println(test());
    }
    static int test() {
        try {
            return 10;
        } catch (Exception e) {
            return 20;
        } finally {
            return 30;
        }
    }
}
```

What will be the output?

- ○ 10
- ○ 20
- [✓ 30] ✓
- ○ Compilation error

✓

```
public class  ChaiPeCharcha  {
  public static void main(String[] args) {
    try {
      validateAge(15);
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
  static void validateAge(int age) throws Exception {
    if (age < 18) {
      throw new Exception("Age must be 18 or above");
    }
    System.out.println("Valid age");
  }
}
```

* 1/1

What will be the output?

- ○ Valid age
- [✓ Age must be 18 or above] ✓
- ○ Exception in thread "main"
- ○ Compilation error

✓ *1/1

```
public class  KhatamTataByeBye   {

    public static void main(String[] args) {

        try {

            String str = args[0];

            int num = Integer.parseInt(str);

            int result = 100 / num;

            System.out.println("Result: " + result);

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.print("No arguments ");

        } catch (NumberFormatException e) {

            System.out.print("Invalid number ");

        } catch (ArithmeticException e) {

            System.out.print("Division by zero ");

        }

    }

}
```

If the program is run with command line argument "0", what will be the output?

- ○ No arguments
- ○ Invalid number
- [✓ Division by zero] ✓
- ○ All of these
- ○ Nothing will be printed

✓
```
public class MujhkoSabAataHai {
    public static void main(String[] args) {
        System.out.println(test());
    }
    static String test() {
        try {
            System.out.print("Try ");
            throw new RuntimeException("Error");
        } catch (RuntimeException e) {
            System.out.print("Catch ");
            return "Catch Return";
        } finally {
            System.out.print("Finally ");
        }
    }
}
```
* 1/1

What will be the output?

- [✓ Try Catch Finally Catch Return] ✓
- Try Catch Catch Return Finally
- Try Finally Catch Return
- Try Catch Finally null

✓

```
public class KhaliDimaag {
    public static void main(String[] args) {
        try {
            try {
                int result = 10 / 0;
            } catch (NullPointerException e) {
                System.out.print("Inner NPE ");
            }
        } catch (ArithmeticException e) {
            System.out.print("Outer AE ");
        }
        System.out.print("End");
    }
}
```

* 1/1

What will be the output?

- ○ Inner NPE End
- [✓ Outer AE End] ✓
- ○ Inner NPE Outer AE End
- ○ End

✓ *

1/1

```
public class OyeHoye {
    public static void main(String[] args) {
        try {
            method1();
        } catch (Exception e) {
            System.out.print("Caught in main");
        }
    }
    static void method1() throws Exception {
        method2();
    }
    static void method2() throws Exception {
        throw new Exception("From method2");
    }
}
```

What will be the output?

- ○ From method2
- [✓ Caught in main] ✓
- ○ Exception in thread "main"
- ○ Compilation error

✗ 0/1

```
public class BadeAccheLagteHo {      *
    public static void main(String[] args) {
        try {
            method();
        } catch (Exception e) {
            System.out.print("Caught: " + e.getMessage());
        }
    }
    static void method() throws Exception {
        try {
            throw new RuntimeException("Try exception");
        } finally {
            throw new Exception("Finally exception");
        }
    }
}
```

What will be the output?

- ○ Caught: Try exception
- ○ Caught: Finally exception
- ○ Both exceptions will be printed
- [✓ Compilation error] ✗

Correct answer

- [✓ Caught: Finally exception]

✓ What is the fundamental difference between Error and Exception in Java's exception hierarchy? *1/1

- ○ Error is for compile-time issues, Exception is for runtime issues
- [✓ Error represents serious problems that applications shouldn't try to catch, Exception represents conditions that applications might want to catch] ✓
- ○ Error is checked, Exception is unchecked
- ○ Error is thrown by user code, Exception is thrown by JVM

✓ Why are checked exceptions called "checked" and what is their primary purpose in Java's design? *1/1

- ○ They are checked at runtime by the JVM for better performance
- [✓ They force the programmer to either handle or declare them at compile time, promoting robust error handling] ✓
- ○ They are automatically checked and handled by Java's garbage collector
- ○ They provide better stack trace information for debugging

✓ What is the primary purpose of the finally block, and when exactly does it execute? *1/1

- ○ To provide an alternative to catch blocks for better performance
- [✓ To ensure cleanup code executes regardless of whether exceptions occur or how the try block exits] ✓
- ○ To handle exceptions that cannot be caught by regular catch blocks
- ○ To log exception information before the program terminates

✓ In what scenarios will a finally block NOT execute? * 1/1

- ○ This is not possible
- ○ When a return statement is present in the try block
- [✓ When System.exit() is called or the JVM crashes] ✓
- ○ When multiple exceptions are thrown simultaneously

✓ Which of the following statements about final in Java is true? * 1/1

- ○ A final variable can be reassigned after initialization
- ○ A final method can be overridden
- [✓ A final class cannot be subclassed] ✓
- ○ A finalize method is automatically called at compile time

✓ What is the purpose of the finally block in Java? * 1/1

- ○ To execute code only when an exception occurs
- [✓ To execute code regardless of whether an exception occurs or not] ✓
- ○ To prevent a class from being subclassed
- ○ To destroy an object immediately

✓ Which statement is TRUE about String objects in Java? * 1/1

- ○ String objects are mutable and can be modified after creation
- ● String objects are immutable and any modification creates a new String object ✓
- ○ String objects can be modified directly using setter methods
- ○ String objects are stored on the stack memory

✓ 1/1

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

What will be the output?

- ○ true, true, true
- ● true, false, true ✓
- ○ false, false, true
- ○ false, true, false

✓ What is the main difference between StringBuilder and StringBuffer? * 1/1

- [✓ StringBuilder is faster but not thread-safe, StringBuffer is slower but thread-safe] ✓
- StringBuilder is thread-safe, StringBuffer is not thread-safe
- StringBuilder works with primitive types, StringBuffer works with objects
- There is no difference between them

✓ StringBuilder sb = new StringBuilder("Java"); * 1/1

```
sb.append(" Programming");

sb.insert(4, " SE");

sb.delete(0, 4);

System.out.println(sb.toString());
```

What will be the output?

- Java SE Programming
- [✓ SE Programming] ✓
- Programming
- SE Java Programming

✓ Which statement about memory usage is correct? * 1/1

- ○ String concatenation using '+' operator is always the most memory efficient
- [✓ ● StringBuilder uses less memory than String for multiple concatenations] ✓
- ○ StringBuffer uses the same amount of memory as String
- ○ All three use exactly the same memory

✓ * 1/1

```
String s1 = "Programming";
String s2 = "Program" + "ming";
String s3 = new String("Programming").intern();
String s4 = new String("Programming");
System.out.println(s1 == s2);
System.out.println(s1 == s3);
System.out.println(s1 == s4);
```

What will be the output?

- ○ true, true, true
- [✓ ● true, true, false] ✓
- ○ false, false, false
- ○ true, false, false

✓ *1/1

```
StringBuffer sb = new StringBuffer("Hello World");

sb.reverse();

sb.replace(0, 5, "Java");

System.out.println(sb.capacity());

System.out.println(sb.length());

System.out.println(sb.toString());
```

Given that initial capacity is typically 16 + string length, what will be printed?

- ○ 27, 9, "Java olleH"
- ○ 16, 9, "Java dlroW"
- [✓ ● 27, 10, "Java olleH"] ✓
- ○ 16, 10, "Java dlroW"

✗ *
0/1

```
public class Test {
    public static void main(String[] args) {
        StringBuilder sb1 = new StringBuilder("abc");
        StringBuilder sb2 = new StringBuilder("abc");


        System.out.println(sb1.equals(sb2));
        System.out.println(sb1 == sb2);
        System.out.println(sb1.toString().equals(sb2.toString()));
    }
}
```

What will be the output?

- [✓ true, false, true] ✗
- false, false, true
- true, true, true
- false, true, false

Correct answer

- [✓ false, false, true]

✓ StringBuilder sb1 = new StringBuilder(); * 1/1

StringBuilder sb2 = new StringBuilder(50);

StringBuilder sb3 = new StringBuilder("Hello");

System.out.println(sb1.capacity());

System.out.println(sb2.capacity());

System.out.println(sb3.capacity());

What will be the output?

- [✓ 16, 50, 21] ✓
- 0, 50, 5
- 16, 50, 16
- 10, 50, 21

✓ StringBuilder sb = new StringBuilder("Java"); * 1/1

String result = sb.append(" is").append(" awesome").reverse().toString();

System.out.println(result);

What will be the output?

- Java is awesome
- [✓ emosewa si avaJ] ✓
- awesome is Java
- avaJ si emosewa

✗ What happens when StringBuilder's buffer capacity is exceeded? * 0/1

- [✓ An exception is thrown] ✗
- The buffer size is automatically doubled
- New characters are ignored
- The buffer size increases by the minimum required amount

Correct answer

- [✓ The buffer size is automatically doubled]

✓ What makes StringBuffer thread-safe? * 1/1

- It uses immutable internal data structures
- [✓ All its methods are synchronized] ✓
- It creates a new object for each operation
- It uses atomic operations

✓ In which scenario should you choose StringBuffer over StringBuilder? * 1/1

- When you need better performance in single-threaded applications
- [✓ When you need thread-safe string operations] ✓
- When you want to use more memory
- When you need immutable strings

✓ class Parent { * 1/1

```
class Parent {
    final void greet() {
        System.out.println("Hello from Parent!");
    }
}
class Child extends Parent {
    void greet() {
        System.out.println("Hello from Child!");
    }
}
public class PapaKehteTheBadaNaamKarega {
    public static void main(String[] args) {
        Child c = new Child();
        c.greet();
    }
}
```

What will happen when you compile and run the above program?

- ○ It will compile successfully and print Hello from Parent!
- ○ It will compile successfully and print Hello from Child!
- [✓ It will fail to compile because the Child class is trying to override a final method] ✓
- ○ It will compile but throw a runtime exception

✓

```
public class EndGame {
    final int a;
    public static void main(String[] args) {
        EndGame obj = new  EndGame();
        System.out.println("Value of a: " + obj.a);
        obj.a = 100;
    }
}
```

* 1/1

What will happen when you compile and run the above program?

- ○ It will compile and print Value of a: 0 and then assign 100 successfully.
- ○ It will compile, but throw a runtime exception when trying to assign 100.
- [✓ It will fail to compile because a final variable cannot be assigned more than once.] ✓
- ○ It will compile and print Value of a: 100.

✓ class BadeChalo { * 1/1

```
class BadeChalo {
    public static void main(String args[])
    {
        int g = 3;
        System.out.print(++g * 8);
    }
}
```

What will be the output?

- [✓ 32] ✓
- 33
- 24
- 25

✓ class YeKyaHai { * 1/1

```
class YeKyaHai {
    public static void main(String args[])
    {
        double a, b,c;
        a = 3.0/0;
        b = 0/4.0;
        c=0/0.0;


       System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

- ○ NaN
- ○ Infinity
- ○ 0.0
- [✓ all of the mentioned] ✓

✓ What will be the output of the following Java program? * 1/1

```
class PureNumberLaunga
{
  public static void main(String args[])
  {
    int x;
    x = 5;
    {
      int y = 6;
      System.out.print(x + " " + y);
    }
    System.out.println(x + " " + y);
  }
}
```

- ● Compilation error ✓
- ○ Runtime error
- ○ 5 6 5 6
- ○ 5 6 5

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

```
class AbbaDabbaJabba
{
  public static void main(String args[])
  {
     if(args.length>0)
     System.out.println(args.length);
  }
}
```

[✓ The snippet compiles and runs but does not print anything] ✓

The snippet compiles, runs and prints 0

The snippet compiles, runs and prints 1

The snippet does not compile

✓ Which of the following is a superclass of every class in Java? * 1/1

ArrayList

Abstract class

[✓ Object class] ✓

String

✓ What will be the output of the following Java program? * 1/1

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

- ○ 2 2
- ○ 3 3
- ○ Runtime Error
- [✓ ● Compilation Error] ✓

✓ Which of this keyword can be used in a subclass to call the constructor of superclass? * 1/1

- [✓ super] ✓
- this
- extent
- extends

✓ What is the process of defining a method in a subclass having same name & type signature as a method in its superclass? * 1/1

- Method overloading
- [✓ Method overriding] ✓
- Method hiding
- None of the mentioned

✓ Which of these is supported by method overriding in Java? * 1/1

- Abstraction
- Encapsulation
- [✓ Polymorphism] ✓
- None of the mentioned

✓ What will be the output of the following Java program? * 1/1

```
class JungleMeMorNachaKisneDekha

{

 public static void main(String[] args)

  {

  int []x[] = {{1,2}, {3,4,5}, {6,7,8,9}};

  int [][]y = x;

  System.out.println(y[2][1]);

  }

}
```

- ○ 2
- ○ 3
- ● 7 ✓
- ○ Compilation Error

✓ What is the value of the variable first after executing the following Java program? *1/1

```
class Abc

{

   public static void main(String[]args)

   {

      String[] elements = { "for", "tea", "too" };

      String first = (elements.length > 0) ? elements[0]: null;

   }

}
```

- ○ Compilation error
- ○ An exception is thrown at run time
- ○ The variable first is set to null
- ● The variable first is set to elements[0] ✓

Experience Zone - No formality 0 of 0 points

Level of Exam * Dropdown

Moderate

I prepare for CCEE Practice Quiz * Dropdown

Yes

Are you enjoying the process. How was your experience (No one word)? *

yes i am enjoying the process. it throws me back to the concepts and makes me doubt myself, which tells i am not thorough with it. i will go through it again.