Working on Constructors and Encapsulation

1. **What will be the output of the following code?**

```
class Base {
    private int x;
    Base(int x) { this.x = x; }
    public int getX() { return x; }
}

class Derived extends Base {
    private int y;
    Derived(int x, int y) {
        super(x);
        this.y = y;
    }
    public int getY() { return y; }
}

public class Test {
    public static void main(String[] args) {
        Derived d = new Derived(10, 20);
        System.out.println(d.getX() + " " + d.getY());
    }
}
```

   - A) `10 20`
   - B) `10`
   - C) `20`
   - D) Compilation error

   Answer: A) `10 20`

2. **Which of the following statements are true about encapsulation in ?**

```
public class Person {
    private String name;
    private int age;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

   - A) The `name` and `age` fields are encapsulated.
   - B) Direct access to `name` and `age` fields is possible outside the class.
   - C) The `getName()` and `getAge()` methods are used to access private fields.
   - D) The `setName()` and `setAge()` methods allow modification of private fields.

   Answer: A, C, D

3. **What will be the output of the following code?**

```
class A {
    A() { System.out.print("A "); }
}

class B extends A {
    B() { System.out.print("B "); }
}

class C extends B {
    C() { System.out.print("C "); }
}

public class Test {
    public static void main(String[] args) {
        new C();
    }
}
```

   - A) `A B C`
   - B) `C B A`
   - C) `A C B`
   - D) `B C A`

   Answer: A) `A B C`

4. **Which of the following will compile without errors?**

```
class Test {
    private Test() {}
    public static Test createInstance() { return new Test(); }
}
```

   - A) `Test t = new Test();`
   - B) `Test t = Test.createInstance();`
   - C) `Test t = Test.createInstance();`
   - D) `Test t = new Test();`

   Answer: B) `Test t = Test.createInstance();`

Code Reusability via Inheritance

5. **What will be the output of the following code snippet?**

```
class Parent {
    void display() { System.out.print("Parent "); }
}
```

```
class Child extends Parent {
    void display() { System.out.print("Child "); }
}

public class Test {
    public static void main(String[] args) {
        Parent p = new Child();
        p.display();
    }
}
```

- A) `Parent`
- B) `Child`
- C) `Parent Child`
- D) `Child Parent`

Answer: B) `Child`

6. **Which code snippet correctly demonstrates method overriding?**

```
class Animal {
    void makeSound() { System.out.println("Animal sound"); }
}

class Dog extends Animal {
    @Override
    void makeSound() { System.out.println("Bark"); }
}

public class Test {
    public static void main(String[] args) {
        Animal a = new Dog();
        a.makeSound();
    }
}
```

- A) The `Dog` class overrides the `makeSound()` method from the `Animal` class.
- B) The `makeSound()` method in `Animal` will not be called.
- C) The `makeSound()` method in `Dog` will be called.
- D) The `Dog` class does not override the `makeSound()` method.

Answer: A, C

Achieving Polymorphism

7. **What will be the output of the following code snippet?**

```
class A {
    void show() { System.out.println("A"); }
}

class B extends A {
    void show() { System.out.println("B"); }
}

public class Test {
    public static void main(String[] args) {
        A obj = new B();
        obj.show();
    }
}
```

- A) `A`
- B) `B`
- C) `A B`
- D) `B A`

Answer: B) `B`

8. **What is true about the following code snippet?**

```
interface Drawable {
    void draw();
}

class Circle implements Drawable {
    public void draw() { System.out.println("Drawing Circle"); }
}

class Square implements Drawable {
    public void draw() { System.out.println("Drawing Square"); }
}

public class Test {
    public static void main(String[] args) {
        Drawable d = new Circle();
        d.draw();
        d = new Square();
        d.draw();
    }
}
```

- A) The `draw()` method of `Circle` is called first, then `Square`.
- B) The `draw()` method of `Square` is called first, then `Circle`.
- C) Both `draw()` methods are called.
- D) The code will not compile.

Answer: A, C

Working on methods of `.lang.Object` class

9. **What will be the output of the following code snippet?**

```
class Person {
    String name;

    Person(String name) { this.name = name; }

    @Override
    public String toString() { return name; }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return
false;
        Person person = (Person) obj;
        return name.equals(person.name);
    }
}

public class Test {
    public static void main(String[] args) {
        Person p1 = new Person("Ajay");
        Person p2 = new Person("Ajay");
        System.out.println(p1.equals(p2));
        System.out.println(p1);
    }
}
```

- A) `true Ajay`
- B) `true Ajay`
- C) `false Ajay`
- D) `false`

Answer: A) `true Ajay`

10. **Which methods are defined in the `Object` class?**
    - A) `clone()`
    - B) `hashCode()`
    - C) `wait()`
    - D) `notify()`
    - E) `equals()`

    Answer: A, B, C, D, E

Object Casting

11. **What is the result of the following code?**

```
class Animal { }
class Dog extends Animal { }

public class Test {
    public static void main(String[] args) {
        Animal a = new Dog();
        Dog d = (Dog) a;
        System.out.println("Cast successful");
    }
}
```

- A) `Cast successful`
- B) `ClassCastException`
- C) `Compilation error`
- D) `Runtime error`

Answer: A) `Cast successful`

12. **Which of the following code snippets are valid for type checking?**

```
Animal a = new Dog();
```

- A) `if (a instanceof Dog) { System.out.println("Dog"); }`
- B) `if (a instanceof Cat) { System.out.println("Cat"); }`
- C) `if (a instanceof Animal) { System.out.println("Animal"); }`
- D) `if (a instanceof Object) { System.out.println("Object"); }`

Answer: A, C, D

Passing Objects as Arguments

13. **What is the output of the following code?**

```
class Box {
    int size;

    Box(int size) { this.size = size; }
}

class Test {
    static void modifyBox(Box b) {
        b.size = 20;
    }

    public static void main(String[] args) {
        Box b = new Box(10);
        modifyBox(b);
        System.out.println(b.size);
    }
}
```

- A) `10`
- B) `20`
- C) `Compilation error`
- D) `Runtime error`

Answer: B) `20`

14. **Which method signatures are valid for a method that takes an `Object` as a parameter?**

```
void process(Object obj);
void process(String str);
void process(int num);
```

- A) `void process(Object obj);`
- B) `void process(String str);`
- C) `void process(int num);`
- D) `void process(Object obj, String str);`

Answer: A, B, C

Abstraction via Abstract Classes and Interfaces

15. **What will be the output of the following code?**

```
abstract class AbstractClass {
    abstract void abstractMethod();

    void concreteMethod() { System.out.println("Concrete Method"); }
}

class ConcreteClass extends AbstractClass {
    void abstractMethod() { System.out.println("Abstract Method"); }
}

public class Test {
    public static void main(String[] args) {
        AbstractClass ac = new ConcreteClass();
        ac.abstractMethod();
        ac.concreteMethod();
    }
}
```

- A) `Abstract Method Concrete Method`
- B) `Concrete Method Abstract Method`
- C) `Abstract Method`
- D) `Concrete Method`

Answer: A) `Abstract Method Concrete Method`

16. **Which of the following statements about interfaces are correct?**

```
interface Animal {
    void eat();
    void sleep();
}
```

```
class Dog implements Animal {
    public void eat() { System.out.println("Dog eating"); }
    public void sleep() { System.out.println("Dog sleeping"); }
}

class Test {
    public static void main(String[] args) {
        Animal a = new Dog();
        a.eat();
        a.sleep();
    }
}
```

- A) `Dog eating` `Dog sleeping`
- B) `Compilation error`
- C) Interfaces cannot have methods with a body.
- D) Interfaces can be used as types for variables.

Answer: A, D

Diamond Problem Using Interfaces

17. **What is the result of the following code?**

```
interface A {
    default void method() { System.out.println("A"); }
}

interface B {
    default void method() { System.out.println("B"); }
}

class C implements A, B {
    public void method() { System.out.println("C"); }
}

public class Test {
    public static void main(String[] args) {
        C c = new C();
        c.method();
    }
}
```

- A) `A`
- B) `B`
- C) `C`
- D) Compilation error

Answer: C) `C`

18. **What will be the output of the following code?**

```
interface X {
    default void show() { System.out.println("X"); }
}

interface Y {
    default void show() { System.out.println("Y"); }
}

class Z implements X, Y {
    public void show() { X.super.show(); }
}

public class Test {
    public static void main(String[] args) {
        Z z = new Z();
        z.show();
    }
}
```

- A) `X`
- B) `Y`
- C) `X Y`
- D) Compilation error

Answer: A) `X`

Creating Static Classes and Static Methods

19. **What will be the output of the following code snippet?**

```
class Outer {
    static int staticVar = 10;

    static class Inner {
        void print() { System.out.println(staticVar); }
    }

    public static void main(String[] args) {
        Inner inner = new Inner();
        inner.print();
    }
}
```

- A) `10`
- B) `Compilation error`
- C) `0`
- D) `null`

Answer: A) `10`

20. **Which of the following statements about static methods are correct?**

```
class Test {
    static void staticMethod() { System.out.println("Static Method");
}

    void instanceMethod() { System.out.println("Instance Method"); }
}

public class Main {
    public static void main(String[] args) {
        Test.staticMethod();
        Test t = new Test();
        t.instanceMethod();
    }
}
```

- A) Static methods can be called without creating an instance of the class.
- B) Instance methods can be called without creating an instance of the class.
- C) Static methods can access instance methods directly.
- D) Static methods cannot access instance variables directly.

Answer: A, D

Wrapper Classes and Autoboxing Concepts

21. **What will be the output of the following code snippet?**

```
Integer x = 10;
Integer y = 10;
System.out.println(x == y);
```

- A) `true`
- B) `false`
- C) `Compilation error`
- D) `Runtime error`

Answer: A) `true`

22. **Which of the following are true about wrapper classes in ?**

```
Integer intObj = 100;
int prim = intObj;
Double dblObj = 10.5;
double primDbl = dblObj;
```

- A) Wrapper classes provide methods to convert to and from primitive types.
- B) Autoboxing and unboxing occur automatically in .
- C) Wrapper classes can hold `null` values.
- D) Wrapper classes are immutable.

Answer: A, B, C, D

Single-Dimensional and Multi-Dimensional Arrays

23. **What will be the output of the following code?**

```
int[][] arr = { {1, 2}, {3, 4} };
System.out.println(arr[1][0]);
```

   - A) `3`
   - B) `4`
   - C) `1`
   - D) `2`

   Answer: A) `3`

24. **Which of the following statements about arrays are correct?**

```
int[] arr = new int[5];
```

   - A) The length of the array is 5.
   - B) The array is initialized with default values (0 for int).
   - C) The array can be resized after creation.
   - D) Array indices start at 0.

   Answer: A, B, D

String Classes

25. **What is the output of the following code snippet?**

```
String str = "";
str = str.concat(" Programming");
System.out.println(str);
```

   - A) ` Programming`
   - B) `Programming`
   - C)
   - D) `Programming`

   Answer: A) ` Programming`

26. **Which of the following statements about `StringBuffer` and `StringBuilder` are correct?**

```
StringBuffer sb = new StringBuffer("Hello");
StringBuilder sb2 = new StringBuilder("World");
```

- A) `StringBuffer` is synchronized, `StringBuilder` is not.
- B) `StringBuilder` is faster than `StringBuffer` due to lack of synchronization.
- C) Both classes are mutable.
- D) `String` is mutable.

Answer: A, B, C

27. **What will be the output of the following code?**

```
String str = "hello";
StringBuilder sb = new StringBuilder(str);
sb.reverse();
System.out.println(sb.toString());
```

- A) `olleh`
- B) `hello`
- C) `null`
- D) `Compilation error`

Answer: A) `olleh`

28. **Which code snippet demonstrates the correct use of regular expressions?**

```
String text = "abc123";
boolean matches = text.matches("\\D+\\d+");
```

- A) `\\D+` matches one or more non-digit characters.
- B) `\\d+` matches one or more digits.
- C) `text.matches("\\D+\\d+")` checks if the text starts with non-digits followed by digits.
- D) `text.matches("\\d+\\D+")` checks if the text starts with digits followed by non-digits.

Answer: A, B, C

29. **What will be the output of the following code?**

```
String regex = "\\d{3}-\\d{2}-\\d{4}";
String str = "123-45-6789";
boolean matches = str.matches(regex);
System.out.println(matches);
```

- A) `true`
- B) `false`
- C) `null`
- D) `Compilation error`

Answer: A) `true`

30. **Which of the following methods are available in `String` class for string manipulation?**

```
String str = "example";
```

- A) `toUpperCase()`
- B) `substring()`
- C) `replace()`
- D) `reverse()`

Answer: A, B, C

Wrapper Classes and Autoboxing Concepts (Additional)

31. **Which of the following statements about autoboxing and unboxing are correct?**

```
Integer obj = 50;
int prim = obj;
```

- A) Autoboxing is the conversion of primitive types to wrapper classes.
- B) Unboxing is the conversion of wrapper classes to primitive types.
- C) Autoboxing and unboxing occur manually in .
- D) Autoboxing and unboxing improve performance by avoiding manual conversion.

Answer: A, B

32. **What will be the output of the following code?**

```
Integer a = 200;
Integer b = 200;
System.out.println(a == b);
```

- A) `true`
- B) `false`
- C) `Compilation error`
- D) `Runtime error`

Answer: B) `false`

33. **What is the output of the following code snippet?**

```
Integer a = 10;
Double b = 10.0;
System.out.println(a.equals(b));
```

    - A) `true`
    - B) `false`
    - C) `Compilation error`
    - D) `Runtime error`

    Answer: B) `false`

34. **Which of the following are true about wrapper classes?**

```
int x = 100;
Integer y = x;
```

    - A) Wrapper classes are immutable.
    - B) Wrapper classes can be used to store primitive values in collections.
    - C) Wrapper classes are synchronized.
    - D) Wrapper classes can be null.

    Answer: A, B, D

Additional Questions on Arrays and Strings

35. **What is the output of the following code snippet?**

```
String[] arr = { "", "Python", "C++" };
System.out.println(arr.length);
```

    - A) `3`
    - B) `2`
    - C) `4`
    - D) `1`

    Answer: A) `3`

36. **Which of the following statements are true about multi-dimensional arrays?**

```
int[][] arr = new int[2][3];
```

- A) `arr` is a 2D array with 2 rows and 3 columns.
- B) `arr[0][0]` is a valid access.
- C) The total number of elements is 6.
- D) The size of the second dimension can vary.

Answer: A, B, C

37. **What will be the output of the following code?**

```
String str = " Programming";
String[] words = str.split(" ");
System.out.println(words[1]);
```

- A) `Programming`
- B)
- C) ` Programming`
- D) `Compilation error`

Answer: A) `Programming`

38. **What will be the output of the following code snippet?**

```
String str = "Hello";
str = str.replace('o', 'a');
System.out.println(str);
```

- A) `Hella`
- B) `Hello`
- C) `Hella`
- D) `Hel`

Answer: A) `Hella`

39. **Which of the following statements about arrays are correct?**

```
int[][] arr = new int[2][];
arr[0] = new int[3];
arr[1] = new int[2];
```

- A) `arr` is a jagged array.
- B) `arr[0]` and `arr[1]` can have different lengths.
- C) All rows in a multi-dimensional array must have the same length.
- D) This code will compile and run successfully.

Answer: A, B, D

40. **What is the output of the following code?**

```
String str = "abc";
StringBuilder sb = new StringBuilder(str);
sb.append("def");
System.out.println(sb.toString());
```

- A) `abcdef`
- B) `abc`
- C) `def`
- D) `Compilation error`

Answer: A) `abcdef`