Single-Dimensional Array, Multi-Dimensional Arrays, Array of Objects, Arrays Utility Class

1. Single-Dimensional Array

1. **Given the following code, what will be the output?**

```
int[] arr = {1, 2, 3, 4, 5};
System.out.println(arr[2]);
```

a) 1

b) 2

c) 3

d) 4

2. Multi-Dimensional Arrays

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

```
int[][] arr = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
System.out.println(arr[2][1]);
```

a) 1

b) 5

c) 6

d) 8

3. Array of Objects

3. **Which of the following code snippets correctly initializes an array of `String` objects?**

```
String[] arr = new String[3];
arr[0] = "Vijay";
arr[1] = "Smitha";
arr[2] = "Ravi";
```

a) Compilation error

b) Runtime error

c) Correct initialization

d) ArrayIndexOutOfBoundsException

4. Arrays Utility Class

4. **Given the following code, what will be the output?**

```
int[] arr = {3, 1, 4, 1, 5};
Arrays.sort(arr);
System.out.println(Arrays.binarySearch(arr, 4));
```

a) 0

b) 2

c) 3

d) -1

String Classes

1. String Class

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

```
String s1 = "hello";
String s2 = new String("hello");
System.out.println(s1 == s2);
```

a) true

b) false

c) Compilation error

d) Runtime error

2. StringBuffer class

6. **Which method of the `StringBuffer` class is used to reverse the characters in the buffer?**

   a) reverse()

   b) invert()

   c) flip()

   d) mirror()

3. StringBuilder class

7. **Given the following code snippet, what will be the output?**

```
StringBuilder sb = new StringBuilder("Hello");
sb.insert(1, "");
System.out.println(sb);
```

   a) Hello

   b) Hello

   c) Hello

   d) Hello

4. Introduction to Regex (Regular Expression)

8. **What does the regex pattern `\d{3}-\d{2}-\d{4}` match?**

   a) Any sequence of 3, 2, and 4 digits separated by hyphens.

   b) Any sequence of digits.

   c) Any sequence of letters.

   d) Any sequence of digits of exactly 9 digits.

Working with Exceptions

1. Defining the Purpose of Exceptions

9. **Which statement is true about the Throwable class?**

   a) It is a checked exception.

   b) It is the superclass of all errors and exceptions in .

   c) It cannot be used directly to create an exception.

   d) It is used to catch all exceptions and errors.

2. Using the try and throw Statements

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

```
try {
    throw new Exception("An error occurred");
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

   a) An error occurred

   b) Compilation error

   c) No output

   d) Runtime error

3. Using the catch, multi-catch, and finally clauses

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

```
try {
    int a = 5 / 0;
} catch (ArithmeticException e) {
    System.out.println("ArithmeticException caught");
} finally {
    System.out.println("Finally block executed");
}
```

   a) ArithmeticException caught

   b) Finally block executed

c) ArithmeticException caught Finally block executed

d) Compilation error

4. Autoclose resources with try-with-resources statement

12. **What is the main benefit of the try-with-resources statement?**

a) It ensures that resources are closed automatically.

b) It allows multiple exceptions to be thrown.

c) It simplifies exception handling.

d) It improves performance.

5. Recognizing Common Exception Classes and Categories

13. **Which of the following is an unchecked exception?**

a) IOException

b) SQLException

c) NullPointerException

d) FileNotFoundException

6. Creating Custom Exceptions

14. **How do you define a custom checked exception in ?**

```
class CustomException extends __ {
    public CustomException(String message) {
        super(message);
    }
}
```

a) RuntimeException

b) Exception

c) Error

d) Throwable

Design Patterns

1. Singleton Design Pattern

15. **Which of the following is a correct implementation of the Singleton pattern?**

```
public class Singleton {
    private static Singleton instance;

    private Singleton() {}

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```

a) Lazy initialization with double-checked locking.

b) Lazy initialization without synchronization.

c) Eager initialization.

d) Static block initialization.

2. Factory Design Pattern

16. **Which statement is true about the Factory design pattern?**

a) It creates objects without specifying the exact class to be instantiated.

b) It provides a way to access a global point of access to an instance.

c) It is used to create a family of related objects without specifying their concrete classes.

d) It ensures a class has only one instance.

3. Abstract Factory Design Pattern

17. **What is the primary purpose of the Abstract Factory pattern?**

a) To create an instance of a class from a specific family.

b) To create instances of related or dependent objects.

c) To allow subclasses to alter the type of objects that will be created.

d) To separate the construction of a complex object from its representation.

4. Builder Design Pattern

18. **Which problem does the Builder pattern primarily solve?**

a) Avoiding the telescoping constructor anti-pattern.

b) Creating an instance of a class from a specific family.

c) Creating objects without specifying the exact class.

d) Ensuring a class has only one instance.

5. Template Method Design Pattern

19. **What is the main purpose of the Template Method pattern?**

a) To define the skeleton of an algorithm in a method, deferring some steps to subclasses.

b) To provide an interface for creating families of related objects.

c) To ensure a class has only one instance.

d) To allow subclasses to alter the type of objects that will be created.

6. Bridge Design Pattern

20. **What does the Bridge pattern help to decouple?**

a) Interface and implementation.

b) Client and service provider.

c) Class hierarchy.

d) Object creation process.

7. Proxy Design Pattern

21. **Which scenario best illustrates the use of the Proxy pattern?**

a) Managing access to a resource by creating a stand-in.

b) Creating an instance of a class from a specific family.

c) Avoiding the telescoping constructor anti-pattern.

d) Defining the skeleton of an algorithm in a method, deferring some steps to subclasses.

8. Creating Immutable Classes

22. **Which of the following is NOT a characteristic of an immutable class in ?**

a) All fields are final.

b) The class is declared as final.

c) The class has setter methods.

d) All fields are private.

8 Features

1. Motivation for Lambdas

23. **What is a primary benefit of lambda expressions in 8?**

a) Improved exception handling.

b) Enhanced performance of primitive operations.

c) Enabling functional programming.

d) Simplified threading.

2. Lambda Expression Overview

24. **Which of the following is a valid lambda expression in 8?**

a) `() -> System.out.println("Hello, World!");`

b) `void() -> System.out.println("Hello, World!");`

c) `() => System.out.println("Hello, World!");`

d) `() -> {return System.out.println("Hello, World!");}`

3. Lambda Expressions and Functional Interfaces

25. **Which of the following is a valid functional interface in 8?**

```
@FunctionalInterface
public interface MyFunction {
    void apply();
}
```

a) Interface with no methods.

b) Interface with a single abstract method.

c) Interface with multiple abstract methods.

d) Interface with multiple default methods.

4. Method References

26. **Which of the following is a valid method reference in 8?**

a) `String::toUpperCase`

b) `Integer->parseInt`

c) `System.out::println`

d) `Math#max`

Working with the Date/Time API

1. The Date/Time API (JSR 310)

27. **What is the primary class in the new Date/Time API for representing a date without a time-zone?**

a) Date

b) Calendar

c) LocalDate

d) ZonedDateTime

2. Use of LocalDate/LocalTime/LocalDateTime Instances

28. **How do you create an instance of `LocalDate` representing the current date?**

a) `LocalDate.now()`

b) `LocalDate.today()`

c) `LocalDate.get()`

d) `LocalDate.new()`

3. Dates and Times across Time Zones

29. **Which class would you use to represent a date and time with a time-zone in the new Date/Time API?**

a) LocalDate

b) LocalTime

c) LocalDateTime

d) ZonedDateTime

4. Formatting Dates

30. **Which of the following is the correct way to format a `LocalDate` object to a string using the `DateTimeFormatter`?**

```
LocalDate date = LocalDate.now();
DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
String formattedDate = date.___formatter);
```

a) toString(formatter)

b) format(formatter)

c) parse(formatter)

d) print(formatter)

Generic Classes

1. Inheritance with Generic Types

31. **Which statement about generic types and inheritance is true?**

    a) Generic types can only inherit from non-generic types.

    b) A generic type can inherit from another generic type with the same type parameter.

    c) Generic types cannot be used with inheritance.

    d) Generic types must implement all methods of the inherited class.

2. Wildcard Parameter Types (Bounded & Unbounded)

32. **Which of the following statements correctly uses an upper-bounded wildcard?**

    a) `List<? extends Number> list = new ArrayList<Integer>();`

    b) `List<? super Number> list = new ArrayList<Object>();`

    c) `List<?> list = new ArrayList<Integer>();`

    d) `List<Number> list = new ArrayList<? extends Number>();`

Programming Fundamentals

1. Creating Primitive Variables

33. **Which of the following is a valid way to declare a primitive variable in ?**

    a) `int a = 5;`

    b) `Integer a = new Integer(5);`

    c) `int a = new int(5);`

    d) `Integer a = 5;`

2. Using Operators

34. **What will be the result of the following expression?**

```
int a = 10;
int b = 20;
int c = a++ + --b;
System.out.println(c);
```

a) 30

b) 29

c) 28

d) 31

3. Using if-else and switch Statements

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

```
int x = 3;
switch(x) {
    case 1: System.out.println("One"); break;
    case 2: System.out.println("Two"); break;
    case 3: System.out.println("Three"); break;
    default: System.out.println("Default");
}
```

a) One

b) Two

c) Three

d) Default

4. Iterating with Loops: while, do-while, for, enhanced for

36. **Which loop is guaranteed to execute at least once?**

a) for

b) while

c) do-while

d) enhanced for

5. Wrapper Classes and Autoboxing Concepts

37. **What is the primary benefit of autoboxing in ?**

a) Improved performance.

b) Simplified code when working with collections.

c) Automatic memory management.

d) Enhanced exception handling.

6. Using Wrapper Classes

38. **Which of the following is the correct way to convert a `String` to an `int`?**

a) `Integer.parseInt("123")`

b) `Integer.valueOf("123")`

c) `Integer.toString("123")`

d) `String.toInt("123")`

7. Keywords

39. **Which keyword is used to define a constant in ?**

a) final

b) static

c) const

d) constant

8. Primitive Data Types

40. **Which of the following is NOT a primitive data type in ?**

a) int

b) boolean

c) String

d) char

OOP Concepts

1. Achieving Encapsulation

41. **Which of the following is NOT a benefit of encapsulation?**

a) Improved code maintainability.

b) Enhanced security.

c) Simplified debugging.

d) Increased performance.

2. Code Reusability via Inheritance

42. **What is the primary advantage of inheritance?**

a) Improved performance.

b) Code reusability.

c) Simplified debugging.

d) Enhanced security.

3. Achieving Polymorphism

43. **Which of the following best describes polymorphism?**

a) One class inheriting from another.

b) One interface implemented by multiple classes.

c) The ability of different classes to be treated as instances of the same class through inheritance.

d) Encapsulating fields in a class.

4. Working on Methods of .lang.Object Class

44. **Which method from the `.lang.Object` class is used for cloning an object?**

a) clone()

b) copy()

c) duplicate()

d) create()

5. Object Casting

45. **Which type of casting is illustrated by the following code?**

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

a) Upcasting

b) Downcasting

c) Implicit casting

d) Autoboxing

6. Passing Objects as Arguments

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

```
class Box {
    int size;
    Box(int size) {
        this.size = size;
    }
}
public class Main {
    public static void main(String[] args) {
        Box box = new Box(10);
        modifyBox(box);
        System.out.println(box.size);
    }
    static void modifyBox(Box box) {
        box.size = 20;
    }
}
```

a) 10

b) 20

c) Compilation error

d) Runtime error

7. Abstraction via Abstract Classes and Interfaces

47. **Which of the following statements is true about abstract classes and interfaces?**

a) Abstract classes cannot have concrete methods.

b) Interfaces cannot have default methods.

c) An abstract class can implement an interface.

d) An interface can have instance variables.

8. Diamond Problem using Interfaces

48. **How does 8 resolve the diamond problem with interfaces?**

a) By using default methods in interfaces.

b) By not allowing multiple inheritance.

c) By implementing classes overriding default methods.

d) By using abstract classes instead of interfaces.

9. Creating Static Classes and Static Methods

49. **Which of the following is NOT true about static methods?**

a) They can access instance variables.

b) They can be called without creating an object of the class.

c) They belong to the class rather than an instance of the class.

d) They can access static variables.

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

```
int[] arr = new int[5];
arr[0] = 1;
arr[1] = arr[0] + 1;
arr[2] = arr[1] * 2;
arr[3] = arr[2] / 3;
arr[4] = arr[3] - 4;
System.out.println(Arrays.toString(arr));
```

a) `[1, 2, 4, 1, -3]`

b) `[1, 2, 4, 1, -4]`

c) `[1, 2, 4, 1, -5]`

d) `[0, 1, 2, 3, 4]`