

### ### 1. String Comparison with == Operator

\*\*Question:\*\* Two strings s1 = "Code" and s2 = new String("Code"); when compared with ==, what is the result?

\*\*Options:\*\*

- A. Compilation error
- B. True
- C. Runtime error
- D. False

\*\*Correct Answer:\*\* D. False

\*\*Explanation:\*\* The == operator compares references (memory addresses) in Java, not the content of strings. s1 is a string literal from the string pool, while s2 is a new object created with the new keyword, so they point to different memory locations despite having the same value. Use .equals() for content comparison.

### ### 2. Enclosure for String Literals in Java

\*\*Question:\*\* In Java, string literals are enclosed within

\*\*Options:\*\*

- A. Angle brackets
- B. Double quotes
- C. Single quotes
- D. Backticks

\*\*Correct Answer:\*\* B. Double quotes

\*\*Explanation:\*\* Java uses double quotes (" ") to define string literals (e.g., String s = "Hello";). Single quotes are for char literals (e.g., char c = 'A';), angle brackets for generics, and backticks are not used for strings in Java.

### ### 3. Custom Exception Type Extending RuntimeException

\*\*Question:\*\* A programmer creates InsufficientBalanceException by extending RuntimeException. What kind of custom exception is this?

\*\*Options:\*\*

- A. Default exception

- B. Checked exception
- C. Error
- D. Unchecked exception

**\*\*Correct Answer:\*\* D. Unchecked exception**

**\*\*Explanation:\*\*** RuntimeException and its subclasses are unchecked exceptions in Java, meaning they do not need to be declared in method signatures or handled explicitly (e.g., via try-catch). Checked exceptions extend Exception (not RuntimeException) and must be handled.

#### ### 4. Concept Applied in Integer Assignment Without new

**\*\*Question:\*\*** A developer writes Integer num = 10, without using new. What concept is applied here?

**\*\*Options:\*\***

- A. Autoboxing
- B. Upcasting
- C. Unboxing
- D. Type casting

**\*\*Correct Answer:\*\* A. Autoboxing**

**\*\*Explanation:\*\*** Autoboxing is the automatic conversion of a primitive type (int 10) to its corresponding wrapper class (Integer). Java handles this implicitly without needing new Integer(10). Unboxing is the reverse (Integer to int), upcasting is for inheritance, and type casting is explicit conversion.

#### ### 5. Result of StringBuilder Append Operation

**\*\*Question:\*\*** A developer uses StringBuilder sb = new  
StringBuilder("Hello").append("World"); What is the result stored in sb?

**\*\*Options:\*\***

- A. Hello World
- B. Hello
- C. World
- D. Error

**\*\*Correct Answer:\*\*** A. Hello World

**\*\*Explanation:\*\*** StringBuilder is mutable and chainable; append("World") adds to the existing "Hello" buffer, resulting in "HelloWorld". (Note: No space is added, but based on the question, it's "HelloWorld". If a space were intended, it would be append(" World"). No error occurs as append returns the StringBuilder instance for chaining.

**### 6. Output of HashMap get(2)**

**\*\*Question:\*\*** What will be the output of the following code?

```
```java
import java.util.*;

public class TestHashMap {
    public static void main(String[] args) {
        HashMap<String, String> map = new HashMap<>();
        map.put("Java", "Language");
        map.put("Python", "Language");
        map.put("JS", "Script");
        System.out.println(map.get(2));
    }
}
```
```

```

**\*\*Options:\*\***

- A. Java
- B. Python
- C. null
- D. JavaScript

**\*\*Correct Answer:\*\*** C. null

**\*\*Explanation:\*\*** HashMap keys are strings here ("Java", "Python", "JS"), but get(2) uses an integer 2 as the key, which doesn't match any string key. Thus, it returns null. Keys are case-sensitive and type-specific; no automatic conversion occurs.

### ### 7. Purpose of StringBuffer Class

**\*\*Question:\*\*** The StringBuffer class in Java is used to create \_\_\_\_ strings.

**\*\*Options:\*\***

- A. Constant
- B. Mutable
- C. Final
- D. Immutable

**\*\*Correct Answer:\*\*** B. Mutable

**\*\*Explanation:\*\*** StringBuffer allows modification of strings after creation (e.g., append, insert), unlike immutable String. It's thread-safe due to synchronization. Constant/final refers to unchangeable references, not the class purpose.

### ### 8. Chronological Order for Creating a Simple HTML Document

**\*\*Question:\*\*** Arrange the given steps in correct chronological order to create a simple HTML document.

**\*\*Steps:\*\***

- A. Add content inside body tag for webpage display
- B. Start with <!DOCTYPE> tag for document type declaration
- C. Write the <html> tag to begin the document
- D. Close the <html> tag to end the document
- E. Use <head> tag to include metadata and title

**\*\*Options:\*\***

- A. A → B → C → D → E
- B. B → C → E → A → D
- C. C → B → E → D → A
- D. E → A → B → C → D

**\*\*Correct Answer:\*\*** B. B → C → E → A → D

**\*\*Explanation:\*\*** HTML5 structure starts with <!DOCTYPE html> (B) for standards mode, then <html> (C) as root, <head> (E) for metadata/title, <body> content (A), and </html> (D) to close. This ensures valid parsing by browsers.

### ### 9. Correct Way to Create a Line Break in HTML

\*\*Question:\*\* Which is the correct way to create a line break in HTML?

\*\*Options:\*\*

- A. <br>
- B. <lb>
- C. <br/>
- D. <lb/>

\*\*Correct Answer:\*\* A. <br>

\*\*Explanation:\*\* <br> is the standard self-closing tag for a line break in HTML5. <br/> is XHTML-compatible but optional. <lb> and <lb/> are invalid and won't render.

### ### 10. HTML Tag for Page Title in Browser Tab

\*\*Question:\*\* A developer wants to define the page title that appears in the browser tab. Which head tag should they use?

\*\*Options:\*\*

- A. <link>
- B. <meta>
- C. <script>
- D. <title>

\*\*Correct Answer:\*\* D. <title>

\*\*Explanation:\*\* <title> inside <head> sets the document title shown in tabs/search results. <link> links resources (e.g., CSS), <meta> provides metadata (e.g., charset), <script> embeds/runs JavaScript.

### ### 11. HTML Tag for Numbered List

\*\*Question:\*\* Which HTML tag is used to create a numbered list?

\*\*Options:\*\*

- A. <ul>
- B. <ol>

C. <li>

**\*\*Correct Answer:\*\*** B. <ol>

**\*\*Explanation:\*\*** <ol> (ordered list) renders numbered items (1., 2.). <ul> is for bulleted unordered lists, <li> defines items within <ol> or <ul>.

### ### 12. CSS Property for Text Color

**\*\*Question:\*\*** Which CSS property is used to change the text color of an element?

**\*\*Options:\*\***

- A. color
- B. background-color
- C. font-color
- D. text-color

**\*\*Correct Answer:\*\*** A. color

**\*\*Explanation:\*\*** color sets foreground text color (e.g., p { color: red; }). background-color affects the element's background. font-color and text-color are non-standard/invalid.

### ### 13. HTML Tag for Table Headers

**\*\*Question:\*\*** In HTML, tables are created using the <table> tag. Rows are defined with <tr>, and table data is placed inside <td>. Which tag is used to start creating table headers that are bold and centered by default?

**\*\*Options:\*\***

- A. <th>
- B. <tr>
- C. <td>

**\*\*Correct Answer:\*\*** A. <th>

**\*\*Explanation:\*\*** <th> creates bold, centered header cells (e.g., <tr><th>Name</th></tr>). <tr> defines rows, <td> is for non-bold, left-aligned data cells.

### ### 14. Standard HTML Document Structure

**\*\*Question:\*\*** A developer is building a webpage and needs to declare the document type and wrap all content in a root element. Which structure should they use?

**\*\*Options:\*\***

- A. <html><body>...</body></html>
- B. <!DOCTYPE html><html><head>...</head><body>...</body></html>
- C. <head><title>...</title></head>
- D. <body><html>...</html></body>

**\*\*Correct Answer:\*\*** B. <!DOCTYPE

html><html><head>...</head><body>...</body></html>

**\*\*Explanation:\*\*** This is the HTML5 skeleton: <!DOCTYPE html> declares type, <html> roots everything, <head> for metadata, <body> for content.

### ### 15. HTML Image Tag with Alternative Text

**\*\*Question:\*\*** A developer wants to display an image on a webpage with alternative text if the image fails to load. Which tag and attribute should they use?

**\*\*Options:\*\***

- A. 
- B. 
- C. <img file="image.jpg" text="description">
- D. 

**\*\*Correct Answer:\*\*** A. 

**\*\*Explanation:\*\*** src specifies the image source, alt provides accessible text for screen readers or load failures. title is for tooltips, file/desc are invalid attributes.

### ### 16. OOP Concept in Class Hierarchy

**\*\*Question:\*\*** A developer creates a Car class and then defines a SportsCar class that uses Car's fields and methods. Which concept is applied here?

**\*\*Options:\*\***

- A. Encapsulation
- B. Abstraction

C. Polymorphism

D. Inheritance

**\*\*Correct Answer:\*\* D. Inheritance**

**\*\*Explanation:\*\*** Inheritance allows SportsCar to extend Car (e.g., class SportsCar extends Car), reusing its fields/methods. Encapsulation bundles data/methods privately, abstraction hides details, polymorphism enables method overriding.

### 17. Symbol for Instance Initializer Block in Java

**\*\*Question:\*\*** Which symbol is used to define an instance initializer block in Java?

**\*\*Options:\*\***

A. { }

B. ()

C. [ ]

D. ;

**\*\*Correct Answer:\*\* A. { }**

**\*\*Explanation:\*\*** Instance initializers are code blocks { ... } outside methods, run when objects are created (e.g., for initialization before constructors). () for parameters, [ ] for arrays, ; terminates statements.

### 18. Output of Static and Instance Method Call

**\*\*Question:\*\*** Consider the following Java code:

```
```java
class Test {
    static void show() { System.out.print("Static"); }
    void display() { System.out.print("Instance"); }
    public static void main(String[] args) {
        new Test().display();
    }
}
````
```

What will be the output?

\*\*Options:\*\*

- A. Instance
- B. Static
- C. Compilation error
- D. Static Instance

\*\*Correct Answer:\*\* A. Instance

\*\*Explanation:\*\* new Test() creates an instance, and display() is non-static, so it calls the instance method, printing "Instance". Static methods (show) don't need instances but aren't called here.

### ### 19. Constructor Usage Without Explicit Definition

\*\*Question:\*\* A programmer creates a class without writing any constructor, yet objects of the class can be created. Which constructor is being used?

\*\*Options:\*\*

- A. Private constructor
- B. Copy constructor
- C. Parameterized constructor
- D. Default constructor

\*\*Correct Answer:\*\* D. Default constructor

\*\*Explanation:\*\* If no constructor is defined, Java provides a public no-arg default constructor (e.g., public MyClass() {}). Private/copy/parameterized must be explicitly written.

### ### 20. Chronological Order for Creating and Using a String in Java

\*\*Question:\*\* Arrange the given steps in correct chronological order to create and use a String in Java:

- A. Save and compile the Java program successfully
- B. Declare a String variable in the program
- C. Run the program to see the String output
- D. Assign a value to the String variable

E. Call String methods like length() or toUpperCase()

\*\*Options:\*\*

- A. A-B-C-D-E
- B. B-D-E-A-C
- C. C-D-B-A-E
- D. E-A-B-D-C

\*\*Correct Answer:\*\* B. B-D-E-A-C

\*\*Explanation:\*\* Declare (B, e.g., String s;), assign (D, s = "Hello";), use methods (E, s.length();), save/compile (A), then run (C) to output.

### ### 21. Purpose of StringBuffer Class (Repeated)

\*\*Question:\*\* The StringBuffer class in Java is used to create \_\_\_\_\_ strings.

\*\*Options:\*\*

- A. constant
- B. mutable
- C. final
- D. immutable

\*\*Correct Answer:\*\* B. mutable

\*\*Explanation:\*\* StringBuffer enables in-place string modifications (e.g., append), unlike immutable String. It's synchronized for thread safety.

### ### 22. Block Always Executed in Exception Handling

\*\*Question:\*\* Which block in Java is always executed whether an exception occurs or not?

\*\*Options:\*\*

- A. catch
- B. try
- C. finally
- D. throw

**\*\*Correct Answer:\*\*** C. finally

**\*\*Explanation:\*\*** finally runs cleanup code after try (or catch if exception), regardless of exception occurrence (e.g., closing resources). catch only if exception matches, try may skip if no exception.

### ### 23. Access Rule for Protected Members

**\*\*Question:\*\*** A developer marks a method as protected. A subclass in another package tries to access it, and it works. Which rule allowed this?

**\*\*Options:\*\***

- A. Protected members are accessible from anywhere
- B. Protected members are only accessible within the same package
- C. Protected members act as public members for subclasses
- D. Protected members are accessible only if the subclass is in the same package

**\*\*Correct Answer:\*\*** C. Protected members act as public members for subclasses

**\*\*Explanation:\*\*** Protected allows access within package or by subclasses (even cross-package via inheritance). Subclasses inherit as if public for their instances.

### ### 24. HTML Semantic Element for Self-Contained Content

**\*\*Question:\*\*** A developer wants to mark up a self-contained piece of content like a news article in HTML. Which new element should they use?

**\*\*Options:\*\***

- A. <div>
- B. <section>
- C. <article>
- D. <aside>

**\*\*Correct Answer:\*\*** C. <article>

**\*\*Explanation:\*\*** <article> represents standalone content (e.g., blog post) for syndication/SEO. <div> is non-semantic, <section> groups thematic content, <aside> for sidebars.

### ### 25. Matching HTML List Types

**\*\*Question:\*\*** Match the type of list in Column A with its correct meaning in Column B.

| Column A          | Column B                                         |
|-------------------|--------------------------------------------------|
| A. Ordered List   | 1. Displays items with bullets (•, o)            |
| B. Unordered List | 2. Displays items with numbers or letters (1, a) |
| C. <ol> tag       | 3. HTML tag used to create an ordered list       |

**\*\*Options:\*\***

- A. A-1, B-2, C-3
- B. A-2, B-1, C-3
- C. A-2, B-1, C-3
- D. A-1, B-3, C-2

**\*\*Correct Answer:\*\*** C. A-2, B-1, C-3

**\*\*Explanation:\*\*** Ordered List (A) uses numbers (2), Unordered List (B) uses bullets (1), <ol> (C) creates ordered lists (3). <ul> is for unordered.

### ### 26. Original Developer of Java

**\*\*Question:\*\*** Java was originally developed by \_\_\_\_\_ at Sun Microsystems.

**\*\*Options:\*\***

- A. James Gosling
- B. Bill Gates
- C. Bjarne Stroustrup
- D. Dennis Ritchie

**\*\*Correct Answer:\*\*** A. James Gosling

**\*\*Explanation:\*\*** James Gosling led the Green Team at Sun Microsystems, releasing Java (originally Oak) in 1995. Others developed C# (Gates/Microsoft), C++ (Stroustrup), C (Ritchie).

### ### 27. Chronological Order for Creating and Using a Class in Java

**\*\*Question:\*\*** Arrange the given steps in correct chronological order to create and use a class in Java:

- A. Compile the Java file using javac command
- B. Create the Java object using new keyword
- C. Write the class with required fields and methods
- D. Save the class with .java extension
- E. Run the file using java command

**\*\*Options:\*\***

- A. A-B-C-D-E
- B. C-D-A-B-E
- C. B-C-D-A-E
- D. E-A-B-C-D

**\*\*Correct Answer:\*\*** B. C-D-A-B-E

**\*\*Explanation:\*\*** Write code (C), save as .java (D), compile to .class (A), instantiate with new (B in code), run (E).

### ### 28. First Official Version of Java

**\*\*Question:\*\*** A developer finds out that the first version of Java was officially released in 1996. Which version was it?

**\*\*Options:\*\***

- A. Java 8
- B. Java SE 5
- C. Java 1.0
- D. Java 2

**\*\*Correct Answer:\*\*** C. Java 1.0

**\*\*Explanation:\*\*** Java 1.0 was released January 23, 1996, by Sun Microsystems. Java 2 (J2SE 1.2) in 1998, SE 5 in 2004, Java 8 in 2014.

### ### 29. Java Concept for Multiple Methods with Same Name

**\*\*Question:\*\*** A developer creates two methods named calculate() in the same class, one taking two integers and another taking two doubles. What Java concept is being used here?

**\*\*Options:\*\***

- A. Constructor chaining
- B. Inheritance
- C. Method overloading
- D. Method overriding

**\*\*Correct Answer:\*\*** C. Method overloading

**\*\*Explanation:\*\*** Overloading allows same-name methods with different parameters (signatures) in the same class. Overriding is for subclasses, chaining for constructors.

### ### 30. Matching HTML Form Elements

**\*\*Question:\*\*** Match the HTML form element in Column A with its correct description in Column B.

| Column A                   | Column B                                 |
|----------------------------|------------------------------------------|
| A. <input type="text">     | 1. Lets users select one or more options |
| B. <input type="checkbox"> | 2. A box where users can type text       |
| C. <input type="submit">   | 3. A button to send the form data        |

**\*\*Options:\*\***

- A. A-1, B-2, C-3
- B. A-2, B-1, C-3
- C. A-3, B-1, C-2
- D. A-1, B-3, C-2

**\*\*Correct Answer:\*\*** B. A-2, B-1, C-3

**\*\*Explanation:\*\*** <input type="text"> (A) for text input (2), checkbox (B) for multi-select (1), submit (C) for form submission (3).

### ### 31. Java Concept for Synchronized Keyword

**\*\*Question:\*\*** A developer uses the synchronized keyword so that only one thread updates a shared file at a time. Which Java concept is this?

**\*\*Options:\*\***

- A. Encapsulation
- B. Abstraction
- C. Synchronization
- D. Polymorphism

**\*\*Correct Answer:\*\*** C. Synchronization

**\*\*Explanation:\*\*** synchronized ensures thread safety by allowing only one thread to access a block/method at a time, preventing race conditions on shared resources.

### ### 32. Object Creation in Java

**\*\*Question:\*\*** A developer writes `Car myCar = new Car();` to build a new car instance and call its methods later. What Java concept is shown here?

**\*\*Options:\*\***

- A. Object creation
- B. Encapsulation
- C. Method overloading
- D. Inheritance

**\*\*Correct Answer:\*\*** A. Object creation

**\*\*Explanation:\*\*** new Car() allocates memory and invokes the constructor, creating an object referenced by myCar. This is instantiation in OOP.

### ### 33. Declaring a Single-Dimensional Array in Java

**\*\*Question:\*\*** What is the correct way to declare a single-dimensional array in Java?

**\*\*Options:\*\***

- A. int[] arr = new int[];
- B. int[] arr = new int[5];
- C. int arr = new int[5];

D. int arr[] = new int[5];

\*\*Correct Answer:\*\* B. int[] arr = new int[5];

\*\*Explanation:\*\* This declares an int array of size 5. D is also valid but less preferred. A lacks size, C mismatches types (array to int).

### ### 34. Loop That Executes at Least Once in Java

\*\*Question:\*\* A loop in Java executes the block of code at least once, even if the condition is false.

\*\*Options:\*\*

- A. while
- B. do-while
- C. forEach
- D. for

\*\*Correct Answer:\*\* B. do-while

\*\*Explanation:\*\* do-while runs the body first, then checks condition. while/for check first (may skip), forEach skips empty collections.

### ### 35. Output of Comparison Code in Java

\*\*Question:\*\* What is the output of the following code?

```
```java
public class Test {
    public static void main(String[] args) {
        int a = 10, b = 20;
        System.out.println(a < b);
        System.out.println(a == b);
    }
}
````
```

\*\*Options:\*\*

- A. false true
- B. true false
- C. false false
- D. true true

**\*\*Correct Answer:\*\*** B. true false

**\*\*Explanation:\*\***  $10 < 20$  is true,  $10 == 20$  is false. Printed on separate lines.  $==$  compares values for primitives.

### ### 36. Output of HashMap Code with Overwrite

**\*\*Question:\*\*** What will be the output of the following code?

```
```java
import java.util.*;

public class TestHashMap {

    public static void main(String[] args) {
        HashMap<String, String> map = new HashMap<>();
        map.put("1", "Java");
        map.put("2", "Python");
        map.put("3", "C++");
        map.put("2", "JavaScript"); // Overwrites key "2"
        System.out.println(map.get("2"));
    }
}
```

**\*\*Options:\*\***

- A. Java
- B. Python
- C. null
- D. JavaScript

**\*\*Correct Answer:\*\*** D. JavaScript

**\*\*Explanation:\*\*** HashMap overwrites values for duplicate keys; "2" changes from "Python" to "JavaScript".

### ### 37. Chronological Order for Nested Loops in Java

**\*\*Question:\*\*** Arrange the given steps in correct chronological order to use nested loops in Java.

- A. Run the program to see inner loop output
- B. Write the outer loop with initialization
- C. Place the inner loop inside the outer loop body
- D. Write the inner loop with initialization
- E. Compile the Java program without syntax errors

**\*\*Options:\*\***

- A. A B C D E
- B. A C B D E
- C. B D C E A
- D. C B D E A

**\*\*Correct Answer:\*\*** C. B D C E A

**\*\*Explanation:\*\*** Outer loop (B), inner loop (D), nest (C), compile (E), run (A).

### ### 38. Matching Java Conditional Statements

**\*\*Question:\*\*** Match the Java conditional statement with its correct description.

Column A	Column B Description
A. if	1. Executes a block if a condition is true, otherwise executes another block
B. else	2. Executes a block only if a condition is true
C. else if	3. Tests another condition if previous if was false

**\*\*Options:\*\***

- A. A-1, B-2, C-3

- B. A-2, B-1, C-3
- C. A-3, B-2, C-1
- D. A-1, B-3, C-2
- E. A-2, B-3, C-1

**\*\*Correct Answer:\*\*** B. A-2, B-1, C-3

**\*\*Explanation:\*\*** if (A) executes only if true (2), else (B) for alternative (1), else if (C) for chained checks (3).

### ### 39. Type of Error for Missing Semicolon

**\*\*Question:\*\*** What type of error occurs if you miss a semicolon at the end of a Java statement?

**\*\*Options:\*\***

- A. Semantic error
- B. Runtime error
- C. Logical error
- D. Syntax error

**\*\*Correct Answer:\*\*** D. Syntax error

**\*\*Explanation:\*\*** Missing ; violates Java grammar, caught at compile-time (e.g., "';' expected"). Semantic is type-related, runtime during execution, logical produces wrong results.

### ### 40. Debugging Technique Using Print Statements

**\*\*Question:\*\*** A programmer uses `System.out.println` statements at different parts of the code to trace variable values. Which debugging technique is this?

**\*\*Options:\*\***

- A. JIT compilation
- B. Logging
- C. Print debugging
- D. Exception handling

**\*\*Correct Answer:\*\* C. Print debugging**

**\*\*Explanation:\*\*** Print statements (e.g., `println("x=" + x)`) trace execution informally. Logging uses frameworks, JIT optimizes runtime, exception handling manages errors.

### ### 41. Output of Concatenation Code

**\*\*Question:\*\*** What will be the output of the following code?

```
```java
public class Test {
    public static void main(String[] args) {
        int a = 5;
        String b = "Hello";
        System.out.println(a + " " + b);
    }
}
```
```

```

**\*\*Options:\*\***

- A. 5 Hello
- B. Hello 5
- C. Compilation Error
- D. null

**\*\*Correct Answer:\*\* A. 5 Hello**

**\*\*Explanation:\*\*** + with String concatenates; int 5 becomes "5", then "5 " + "Hello" = "5 Hello".

### ### 42. Arithmetic Operator for Remainder

**\*\*Question:\*\*** Which arithmetic operator in Java is used to find the remainder of a division?

**\*\*Options:\*\***

- A. - (subtraction)
- B. \* (multiplication)
- C. % (modulo)

D. / (division)

\*\*Correct Answer:\*\* C. % (modulo)

\*\*Explanation:\*\* % returns remainder (e.g.,  $10 \% 3 = 1$ ). / gives quotient, -/\* are other ops.