Flynaut SaaS Pvt. Ltd.

Phase I

Puzzle-Based MCQs

Which of the following truly breaks encapsulation?

Options:

A. Using private fields
B. Using setters to update private fields
C. Making fields public
D. Using constructors for initialization

You add null twice into a HashSet. What will happen?

Options:

A. Both nulls will be stored
B. Compilation error
C. Only one null will be stored
D. Runtime exception

You add 5 elements to an ArrayList, then use remove(2). What gets removed?

Options:

A. Element at index 2
B. Element with value 2
C. Both
D. Error

Which collection maintains insertion order and allows duplicates?

Options:

A. HashSet

B. TreeSet

C. LinkedHashSet

D. ArrayList

Which statement is *true* about Set and List?

Options:

A. Both allow duplicates

B. List maintains order, Set doesn’t

C. Set maintains order, List doesn’t

D. Both don't allow null

I’m a child of the List family,
I grow as you add, dynamically.
Insertions are my charm, removals no fear,
But my performance drops if index isn’t near.
Who am I?

Options:

A. LinkedList
B. Vector
C. HashSet
D. TreeSet

You add me in order, I stay that way.
Duplicates? Sorry, I don’t let them stay.
I’m not sorted, but I’m ordered for sure.
If insertion matters, I’m your cure.
Who am I?

Options:

A. HashSet
B. TreeSet
C. LinkedHashSet
D. ArrayList

I am used when you want both uniqueness and sorting.
I organize elements in natural order
But remember—I don’t allow null.
Who am I?

Options:

A. HashSet
B. TreeSet
C. ArrayList
D. LinkedList

You can use me to refer to the current object.
I help you resolve name clashes between instance variables and parameters.
I’m short and powerful—just four letters.
Who am I?

Options:

A. self
B. super
C. this
D. base

I’m flexible and grow dynamically,
I allow duplicates and keep order dramatically.
Access me fast by index,
But inserting in the middle is complex.
Who am I?

Options:

A. LinkedList
B. TreeSet
C. HashSet
D. ArrayList

Phase II

Guess the Error/Output

```
List<String> list = new ArrayList<>();
list.add("A");
list.add("B");
list.add("A");
Set<String> set = new HashSet<>(list);
System.out.println(set.size());
```

Options:

A. 2

B. 3

C. 1

D. Error

```
abstract class Animal {
    abstract void sound() {}
}
```

Options:

A. No error

B. Abstract method cannot have a body

C. Class must be final

D. sound() must be static

```java
class Test {
    int x;

    Test(int x) {
        this.x = x;
    }

    public static void main(String[] args) {
        Test t = new Test(5);
        System.out.println(t.x);
    }
}
```

Options:

A. 0

B. 5

C. Compile-time error

D. Runtime error

```
Set<String> set = new HashSet<>();
System.out.println(set.get(0));
```

Options:

A. null

B. IndexOutOfBoundsException

C. Compilation Error

D. UnsupportedOperationException

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

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

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

Options:

A. A

B. B

C. A B

D. B A

```java
class Test {
    private Test() {
        System.out.println("Private constructor");
    }

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

Options:
A. Error: Private constructor not allowed
B. Output: Private constructor
C. Error: Cannot instantiate class
D. Error at runtime

```
List<String> list = new ArrayList<>();
list.add("One");
list.add("Two");
list.add("Three");
System.out.println(list.get(3));
```

Options:

A. Three
B. null
C. IndexOutOfBoundsException
D. Error

```
Set<String> set = new HashSet<>();
set.add(null);
set.add(null);
System.out.println(set.size());
```

Options:

A. 0

B. 1

C. 2

D. Error

```java
List<Integer> list = new LinkedList<>();
list.add(100);
list.add(200);
list.add(0, 50);
System.out.println(list);
```

Options:

A. [100, 200, 50]

B. [50, 100, 200]

C. [200, 100, 50]

D. Error

```java
Set<String> set = new HashSet<>();
set.add("Apple");
set.add("Banana");
set.add("apple");
System.out.println(set.size());
```

Options:

A. 2

B. 3

C. 1

D. Error

```java
class A {
    private void display() {
        System.out.println("A");
    }
}
class B extends A {
    private void display() {
        System.out.println("B");
    }
}
public class Test {
    public static void main(String[] args) {
        B b = new B();
        b.display();
    }
}
```

Options:

A. A

B. B

C. Compile error

D. Method hiding, not overriding

Phase III

Coding Questions

Problem Statement:

Remove Duplicates Using List

Write a program that takes a list of integers with duplicates and returns a list without duplicates, preserving insertion order.

Input:

[10, 20, 10, 30, 20, 40]

Expected Output:

[10, 20, 30, 40]

Problem Statement:

Unique Word Counter

Take a string input from the user, split it into words, and print the count of unique words using a Set.

Example Input:

"Java is fun and Java is powerful"

Expected Output:

Unique words count: 5

Problem Statement:

Student Score Manager
Create a Student class with fields id, name, and score.
Add multiple student objects to an ArrayList.

Now write logic to:
Print students who scored more than 80

Remove students who scored below 40

Print the final list

Problem Statement:

Reverse a List

Write a program to reverse a list of integers without using Collections.reverse().

Input:

[1, 2, 3, 4, 5]

Expected Output:

[5, 4, 3, 2, 1]

Problem Statement:

Employee Role Filter

Create an Employee class with id, name, and role.
Add multiple employees to a HashSet.
Print all employees whose role is "Developer".

Problem Statement:

Insert Element at Every Even Index

Given a list of strings, insert the word "Hello" at every even index (0, 2, 4...) and print the result.

Input:

["Java", "Python", "C++"]

Expected Output:

["Hello", "Java", "Hello", "Python", "Hello", "C++"]