Java Interview Q&A with Explanations and System

Q: What are the four pillars of OOP in Java?

✓ Interview Answer:

The four pillars of OOP are Encapsulation, Abstraction, Inheritance, and Polymorphism.

- **Explanation**:
- 1. **Encapsulation**: Wrapping variables and methods together in a class, restricting direct access.
- 2. **Abstraction**: Hiding implementation details and showing only necessary features using abstract classes/interfaces.
- 3. **Inheritance**: A class can acquire properties and methods of another class using 'extends' keyword.
- 4. **Polymorphism**: Ability of a method or object to take many forms. (Method overloading/overriding).

Q: What is an Interface in Java?

✓ Interview Answer:

An interface is a blueprint of a class that contains abstract methods and constants. It is used to achieve abstraction and multiple inheritance.

Explanation:

Interfaces define what a class should do, but not how it does it. A class implements an interface and provides functionality.

Example:

interface Animal { void sound(); }
class Dog implements Animal { public void sound() {
 System.out.println("Bark"); } }

Q: Difference between Abstraction and Interface

✓ Interview Answer:

Abstraction focuses on hiding implementation using abstract classes, whereas Interface provides a contract of methods a class must implement.

Explanation:

Abstract class can have both abstract and concrete methods, while interface only had abstract methods (until Java 8 added default/static).

Q: Difference between Abstraction and Encapsulation

✓ Interview Answer:

Abstraction hides implementation details, while Encapsulation hides the internal state using access modifiers.

Explanation:

Encapsulation = data hiding (fields private, accessed via getters/setters). Abstraction = design-level hiding of implementation using abstract classes/interfaces.

Q: What is static in Java?

✓ Interview Answer:

The 'static' keyword means the member belongs to the class rather than an instance.

Explanation:

Static variables are shared across objects, static methods can be called without creating an object, and static blocks are executed once when class loads.

Example:

```
class Test {
    static int count = 0;
    static void show(){ System.out.println(count); }
    static { System.out.println("Static block executed first"); }
}
```

Q: Can static block run first or later?

✓ Interview Answer:

Static block runs first, before the main method, when the class is loaded into memory.

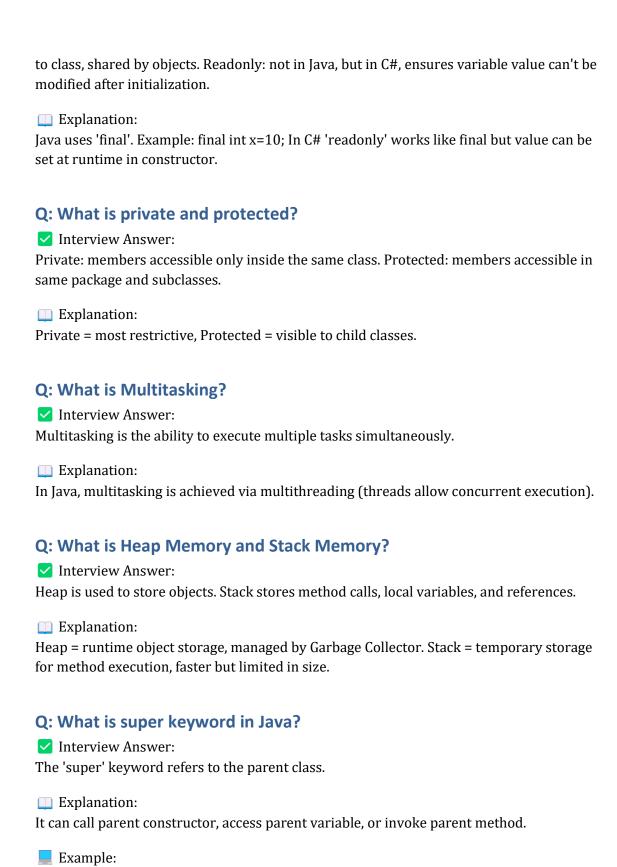
Explanation:

It initializes static variables or performs setup tasks before object creation.

Q: Difference between readonly, final, and static

Interview Answer:

Final: used to declare constants, prevent method overriding and inheritance. Static: belongs



```
class Animal { void sound() { System.out.println("Animal sound"); } }
class Dog extends Animal {
   void sound() { super.sound(); System.out.println("Dog barks"); }
}
```

Mini Library Management System (Covers OOP Pillars)

This system demonstrates all four OOP pillars:

- **Encapsulation**: private fields with getters/setters.
- **Abstraction**: abstract class Book with abstract method.
- **Inheritance**: different types of Books extend Book.
- **Polymorphism**: overriding displayDetails() method.

```
abstract class Book {
  private String title;
  private String author;
 public Book(String title, String author) {
   this.title = title;
    this.author = author;
 }
  public String getTitle() { return title; }
  public String getAuthor() { return author; }
 public abstract void displayDetails();
class TextBook extends Book {
 private String subject;
  public TextBook(String title, String author, String subject) {
    super(title, author);
    this.subject = subject;
  }
  @Override
  public void displayDetails() {
    System.out.println("TextBook: " + getTitle() + " by " +
```

```
getAuthor() + ", Subject: " + subject);
}
class Novel extends Book {
  private String genre;
  public Novel(String title, String author, String genre) {
    super(title, author);
    this.genre = genre;
  @Override
  public void displayDetails() {
    System.out.println("Novel: " + getTitle() + " by " + getAuthor() +
", Genre: " + genre);
}
public class LibrarySystem {
  public static void main(String[] args) {
    Book b1 = new TextBook("Java Basics", "James Gosling",
"Programming");
    Book b2 = new Novel("Sherlock Holmes", "Arthur Conan Doyle",
"Mystery");
    b1.displayDetails();
    b2.displayDetails();
  }
```