Here’s a **detailed explanation and code examples** for the FAQs:

**1. FAQs on Java Basics**

**1. What is Java?**  
Java is a high-level, general-purpose, object-oriented programming language. Its key feature is platform independence, achieved through the Java Virtual Machine (JVM). It is widely used for building web applications, mobile apps, and enterprise software.

**2. What are the main features of Java?**

* **Platform Independence**: Java programs run on any operating system with JVM.
* **Object-Oriented**: Supports OOP concepts like inheritance, encapsulation, polymorphism, and abstraction.
* **Robust**: Provides features like exception handling, strong memory management, and type safety.
* **Secure**: Built-in security mechanisms prevent unauthorized access and program vulnerabilities.
* **Multithreading**: Java supports multiple threads, making it efficient for tasks like games or real-time applications.

**3. What is the difference between JDK, JRE, and JVM?**

* **JDK (Java Development Kit):** A package for developing Java applications, including compiler (javac), debugger, and tools.
* **JRE (Java Runtime Environment):** Provides libraries and JVM needed to run Java programs.
* **JVM (Java Virtual Machine):** Executes the compiled Java bytecode.

**4. What are the main principles of OOP in Java?**

1. **Encapsulation**: Restricting direct access to certain components of an object.
2. **Inheritance**: Acquiring properties of a parent class in a child class.
3. **Polymorphism**: One interface, many implementations (e.g., method overloading/overriding).
4. **Abstraction**: Hiding implementation details and exposing only functionality.

**5. What is a Class and an Object in Java?**

* **Class**: A blueprint for creating objects. It defines attributes (fields) and behaviors (methods).
* **Object**: An instance of a class that holds data and performs tasks.

Example:

class Car {

String brand;

int speed;

void displayInfo() {

System.out.println("Brand: " + brand + ", Speed: " + speed + "km/h");

}

}

public class Main {

public static void main(String[] args) {

Car car1 = new Car();

car1.brand = "Toyota";

car1.speed = 120;

car1.displayInfo();

}

}

**6. What is the difference between == and equals() in Java?**

* ==: Compares memory references, not content.
* equals(): Compares the content of objects.

Example:

String str1 = new String("Hello");

String str2 = new String("Hello");

System.out.println(str1 == str2); // false (different references)

System.out.println(str1.equals(str2)); // true (same content)

**7. What is a constructor in Java?**  
A constructor initializes an object when it is created. It has the same name as the class and no return type.

Example:

class Student {

String name;

// Constructor

Student(String name) {

this.name = name;

}

void display() {

System.out.println("Name: " + name);

}

}

**8. What is the main method in Java?**  
The entry point for any Java application:

public static void main(String[] args) {}

**9. What is a package in Java?**  
A package organizes classes and interfaces into namespaces. Example: java.util, java.io.

**10. What are access modifiers in Java?**  
Modifiers determine scope/visibility:

* public: Accessible anywhere.
* protected: Accessible within the package or subclass.
* default: Accessible within the package.
* private: Accessible only within the class.

**2. FAQs on String Handling**

**1. What is a String in Java?**  
Strings are sequences of characters represented by the String class. Strings are immutable, meaning their values cannot be changed after creation.

**2. How do you create a String in Java?**

1. Using string literals:
2. String str = "Hello";
3. Using the new keyword:
4. String str = new String("Hello");

**3. How do you concatenate Strings?**

String greeting = "Hello, " + "World!";

String message = greeting.concat(" How are you?");

**4. How do you compare Strings?**

* Use equals() to compare values.
* Use equalsIgnoreCase() for case-insensitive comparison.
* Use compareTo() for lexicographical comparison.

**5. Difference between == and equals()?**

* ==: Reference comparison.
* equals(): Content comparison.

**6. Convert String to Uppercase or Lowercase:**

String upper = str.toUpperCase();

String lower = str.toLowerCase();

**7. Extract a Substring:**

String sub = str.substring(2, 5); // Characters from index 2 to 4

**8. Check if a String contains a substring:**

boolean contains = str.contains("abc");

**9. Replace characters in a String:**

String replaced = str.replace('a', 'o');

**3. FAQs on Classes and Objects**

**1. Access Modifiers:**

* public, protected, private, default.

**2. Difference between Abstract Class and Interface:**

* Abstract classes can have concrete methods; interfaces (pre-Java 8) only have abstract methods.
* A class can implement multiple interfaces but can extend only one abstract class.

**3. Difference between Constructors and Methods:**

* Constructors initialize objects; methods define behavior.

**4. What is an Interface?**  
An interface is a contract where implementing classes define methods.

**5. Predefined Packages:**

* java.util, java.io, java.net, java.sql, etc.

**4. FAQs on Inheritance**

**1. Explain Inheritance:**  
Inheritance allows one class to inherit fields and methods of another.

**2. Advantages of Inheritance:**

* Code reuse
* Simplified maintenance
* Enables polymorphism

**3. Parent vs. Child Class:**

* Parent: Base class whose properties are inherited.
* Child: Derived class inheriting from a parent.

**4. Why no multiple inheritance in Java?**  
To avoid ambiguity caused by the Diamond Problem.

**5. Types of Inheritance:**

* Single
* Multilevel
* Hierarchical

**5. FAQs on Abstract Classes**

**1. Difference between Abstract Class and Concrete Class:**

* Abstract: Cannot be instantiated; contains abstract and concrete methods.
* Concrete: Fully implemented, instantiable class.

**2. Explain Abstraction vs Encapsulation:**

* Abstraction hides implementation; Encapsulation binds data and methods.

**3. Static vs Instance Variables:**

* Static: Shared across all objects.
* Instance: Unique to each object.

**4. Java Abstract Class Example:**

abstract class Shape {

int x, y;

abstract void printArea();

}

class Rectangle extends Shape {

void printArea() {

System.out.println("Rectangle Area: " + (x \* y));

}

}

public class Main {

public static void main(String[] args) {

Rectangle rect = new Rectangle();

rect.x = 10;

rect.y = 5;

rect.printArea();

}

}

**5. Advantage of Abstract Class:**  
Abstract classes provide a partial abstraction by allowing some shared implementation while leaving the rest to the subclasses.

Let me know if you need additional examples or explanations for specific topics!