Class # 11

1. **Abstraction in Java:**

Abstraction Concept:

Abstraction is one of the core concepts in Java and object-oriented programming (OOP). It involves simplifying complex reality by modeling classes based on the essential properties and behaviors of objects, while hiding unnecessary details. In other words, abstraction allows you to focus on what an object does rather than how it does it.

Here's a breakdown of abstraction in Java:

1. **Abstract Classes and Methods**: Abstraction is often implemented in Java using abstract classes and methods. An abstract class is a class declared with the **abstract** keyword, and it may contain abstract methods, which are methods without a body. Abstract methods are meant to be overridden by concrete (non-abstract) subclasses.

java code

abstract class Shape {

abstract void draw(); // Abstract method

}

2. **Extending Abstract Classes**: When you want to create a specific type of object based on an abstract class, you extend that abstract class and provide concrete implementations for its abstract methods.

java code

class Circle extends Shape {

void draw() { // Concrete implementation for drawing a circle

}

}

3. **Abstraction Hierarchy**: You can create an abstraction hierarchy where you have a base abstract class with common attributes and methods, and then concrete subclasses that inherit from it. Each subclass provides specific implementations.

java code

```
abstract class Animal {
abstract void makeSound();
}
class Dog extends Animal {
void makeSound() {
System.out.println("Woof!");
}
}
class Cat extends Animal {
void makeSound() {
System.out.println("Meow!");
}
}
```

4. **Interfaces**: Java also supports abstraction through interfaces. An interface defines a contract of methods that implementing classes must provide. Interfaces are used to achieve abstraction by defining a set of methods without specifying their implementation.

java code

```
interface Printable {
void print();
}
```

```
class Document implements Printable {

public void print() { // Concrete implementation for printing a document

}

}
```

5. **Benefits of Abstraction**:
    - **Simplification**: Abstraction simplifies complex systems by breaking them down into manageable, understandable pieces.
    - **Reusability**: Abstract classes and interfaces allow you to define reusable templates for subclasses or implementing classes.
    - **Flexibility**: Abstraction provides a level of flexibility that allows you to change the internal implementation of a class without affecting the code that uses the class.

Abstraction is a fundamental concept in OOP and is used extensively in Java to design clean, modular, and maintainable code. It helps you create a clear separation of concerns and promotes code reusability.
