JAVA @11

Abstract Classes

Objective

After completing this session you will be able to understand,

- Introduction to Abstract Classes
- When to use abstract classes

The word - Abstract

Dictionary Meaning of Abstract:

a. *Conceptual*

b. *Theoretical*

c. *Existing in thought or as an idea but not having a physical or concrete existence.*

Abstract Class

Abstract Class in Java

An abstract class is a class that contains one or more abstract methods.

Example:

```
abstract class Calculator{
        public void display() {
        }
        public abstract void calculateVolume();
}
```

- Abstract class should use the keyword "*abstract*"
- An abstract class can contain non abstract methods also.
- If a class has at least one abstract method, then that class should be declared abstract.

Abstract Method

Abstract Method in Java:

Methods that do not have any implementation(body) are called abstract methods

Example:

Abstract Method:

Public abstract void add();

Concrete Method:

```
public void add(){
    int x = a + b;
}
```

To *create* an *abstract method*, just write the method declaration without the body and use the keyword "*abstract*"

DO NOT USE CURLY BRACES.

Extending an Abstract class

Assume a parent class as LivingThing

```
public abstract class LivingThing {

    public abstract void walk();
}
```

Abstract method - walk() in parent class

Human subclass of LivingThing

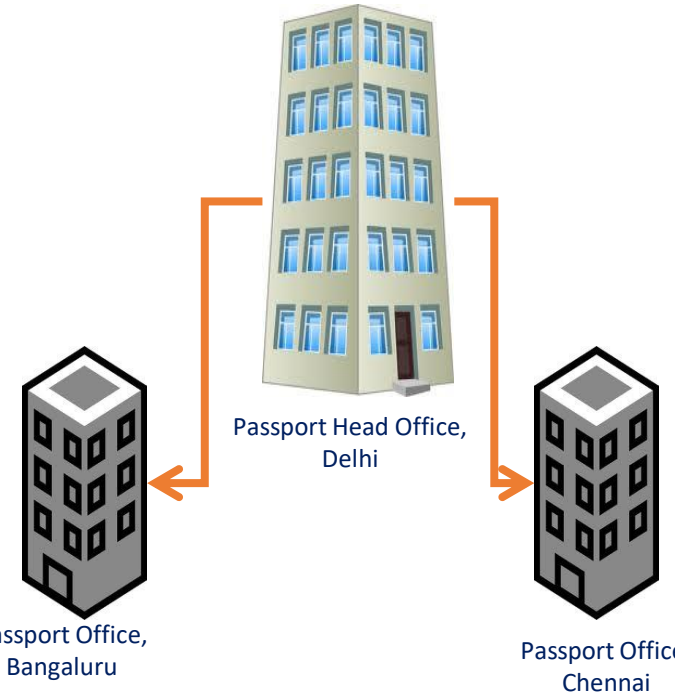
```
public class Human extends LivingThing {

    @Override
    public void walk() {
        System.out.println("Human walks with two legs");
    }
}
```

Implementation of abstract method - walk() in subclass

Abstract in real world

Scenario: Lets take a passport issuance process, assume that the passport head office is located in ***Delhi*** and corresponding local offices are in ***Chennai/Bangaluru***. The passport issuance is a two step process, ***verification of photo identity*** and ***final approval***. In Chennai and Bangalore the verification document for photo identity are different. Say in Bangalore they verify using ration card and in Chennai they use drivers license. So the head office at Delhi decides to define a protocol and delegate the photo identity verification process but the final approval will be done by the head office.

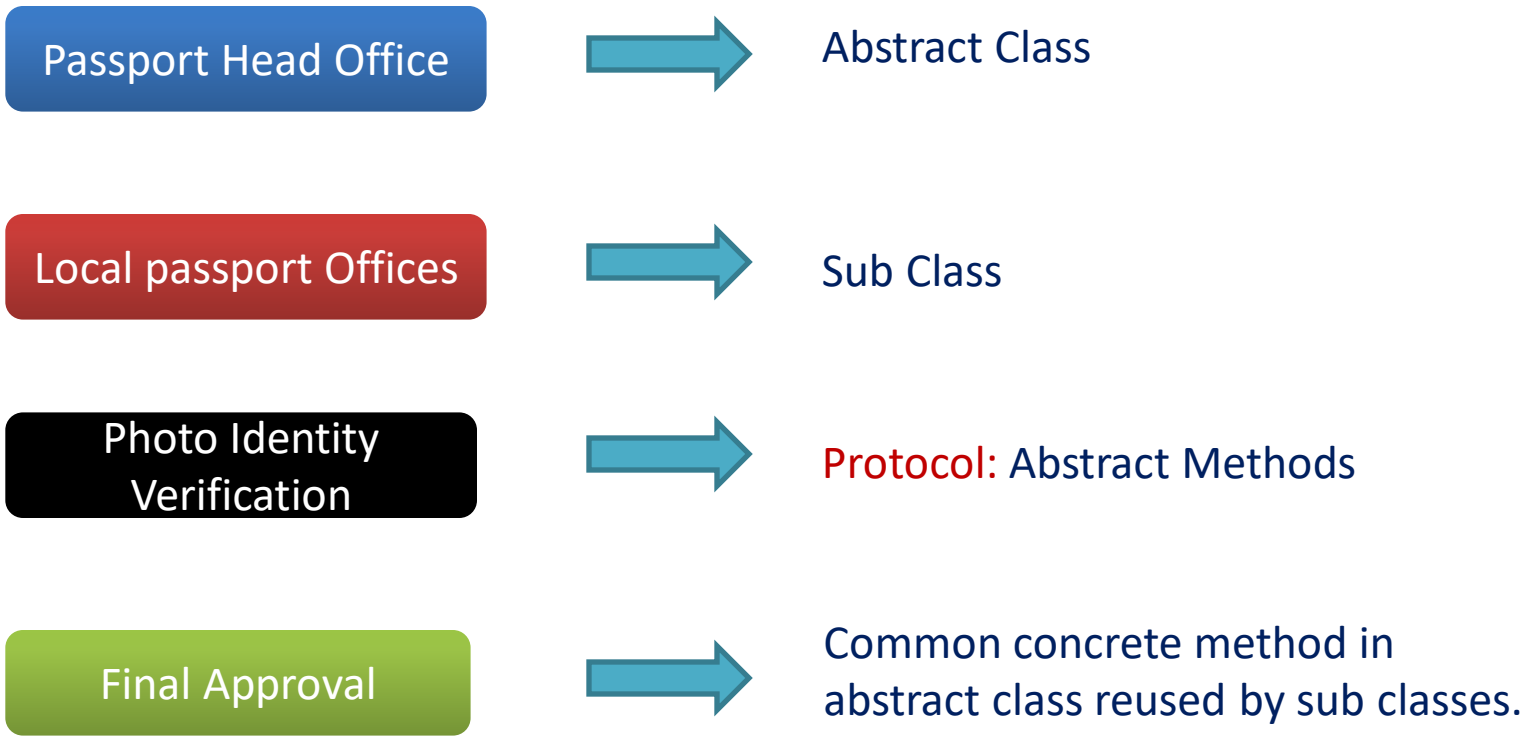
Bangaluru,
- Adheres to the protocol and performs photo identification using ration card.
- Sends the document to HO for approval.



Delhi Office,
- Defines a protocol, that all local passport office should verify the photo identity.
- HO verifies and approves the final document.

Chennai,
- Adheres to the protocol and performs photo identification using driver license.
- Sends the document to HO for approval.

Abstract class & Passport office



NOTE: No people can directly reach Delhi head office for passport application rather they should always go via the local offices in appropriate cities. Similarly abstract Class cannot be instantiated only the subclasses can be instantiated and methods in abstract class invoked.

When to use Abstract classes?

When do we use abstract classes and Methods?

- Abstract class (*Passport Head Office*) is one way of dictating a strict protocol for the subclasses (*Local passport offices*) to follow, that is implement the abstract methods (*verify photo identity*) declared in parent class.

NOTE: Defining methods to be implemented (protocol) cannot be done in case of normal class being extended.

- Abstract methods are usually declared where two or more subclasses are expected to fulfill a similar role (abstract methods) in different ways

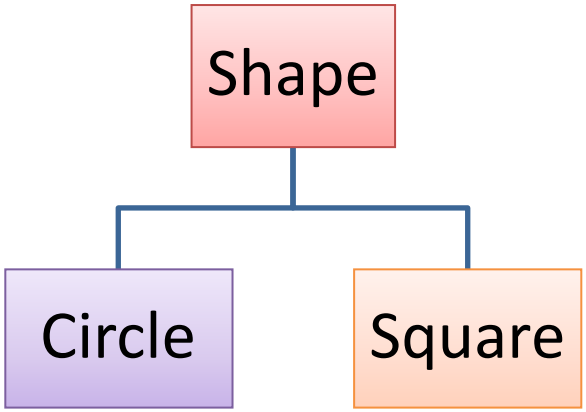
These subclasses extend the same abstract class and provide different implementations for the abstract methods.

- You can use abstract classes to implement common methods (passport final approval) and use the subclasses to implement the abstract methods specific to the sub class.

When to use Abstract classes?

When do we use abstract classes and Methods? – Software Example

Assume Shape is the parent class, Circle and Square are subclasses that extends Shape. All the Shapes should have red color and they should expose a method to calculate area based on their appropriate shape.



Abstract class ***Shape***
- Implements the ***common*** method ***setColor***.
- Declares a abstract method (protocol) ***calculateArea*** for sub class to implement.

Classes ***Square*** & ***Circle*** extend ***Shape*** and
- Inherit ***setColor*** method.
- Implement abstract method ***calculateArea*** (adhere the protocol).

Example – Abstract classes

Let us understand more about abstract classes with the below program

1. Create a parent class **Shape.java** with a instance variable color and methods below,
    a. Abstract method **calculateArea**() which returns double value;
    b. Concrete method **setColor** which accepts a String *color* as the parameter and sets the instance variable ‘color’. It should also print the color in the console.
2. Create a sub class **Circle.java** which extends Shape.java and
    a. Implements the ***calculateArea()*** method. It should calculate the area as 3.14*r*r and print the area in the console. Consider radius = 5.0.
3. Create another sub class **Square.java** which extends Shape and
    a. Implements the ***calculateArea()*** method. It should calculate the area as length*breadth and print the area in the console. Consider Length/Breadth = 4.0.
4. Create ***AbstractDemo.java*** class with Main method which invokes the **setColor** method and the **calculateArea**() method on Circle and Square Object

Example - Solution

```
abstract class Shape{
    String color;
    void getColor(String color) {
        this.color = color;
        System.out.println(color);
    }
    abstract double calculateArea();
}
class Circle extends Shape{
    @Override
    double calculateArea() {
        double radius = 5.0;
        getColor("Circle:: green");
        return 3.14*radius*radius;
    }
}
```

```
class Square extends Shape{
    @Override
    double calculateArea() {
        double area = 4.0;
        getColor("Square:: red");
        return area*area;
    }
}
public class AbstractDemo {
    public static void main(String[] args) {
        Shape obj = new Circle();
        double aoc = obj.calculateArea();
        System.out.println("Area of Circle : "+aoc);

        obj = new Square();
        double aos = obj.calculateArea();
        System.out.println("Area of Square : "+aos);
    }
}
```

Thank you

You have successfully completed

Abstract Class APIs