Abstract Class

Abstract Class

- Abstract class is a facility to collect generic or common features into a single super class.
- One or more methods are declared but not defined.
- Helps to create base classes that are generic and the implementation is not available.

Abstract Class

```
public abstract class Shape {
    public abstract void draw();
}
public class Circle extends Shape {
    public void draw(){
        //Code to draw Circle
    }
}
```

Abstract Class

- Any class that has even one method as abstract, should be declared as abstract.
- Abstract classes cannot be instantiated.
- `abstract` modifier is not applicable for variables, constructors and static methods.

Abstract Class

- Any subclass of an abstract class must implement all the abstract methods; otherwise should be declared as `abstract`.
- An abstract class may contain concrete methods also.

Abstract Class

```
public abstract class Shape {
    public abstract void draw();
    public void erase(){
        //Code to erase the shape
    }
}
public class Circle extends Shape {
    public void draw(){
        //Code to draw the Circle
    }
}
```

Using `final`

Using `final`

- A keyword in Java.
- Can be used to declare variables that are immutable.

Using `final`

- E.g.

```
final float PI = 3.14f;
 PI = 4.0f;//ERROR
```

- Final variables must be initialized.

Using `final`

- If a final variable is a reference to an object, it is the reference that must stay the same and not the object.
- E.g.

```
final City c1 = new City("Pune");
 c1.setName("Mumbai"); //OK
 c1 = new City("Mumbai"); //ERROR
```

Using `final`

- `final` can also be applied for methods to prevent method overriding.
- `private` methods of a class are implicitly final.

Using `final`

- A class can also be declared as `final` to prevent inheritance.
- If so, all the methods of that class become `final`.

Lets Summarize

- Containment
- Inheritance
- Working with `super`
- Method Overriding
- Abstract Classes
- Using `final`