VNR Vignana Jyothi Institute of Engineering and Technology

An autonomous Institute – NAAC 'A' and NBA Accredited

TAMASOMA JYOTIRGAMAYA
Estd. 1995

Department of Computer Science and Engineering

Learning By Doing

On

JAVA PROGRAMMING

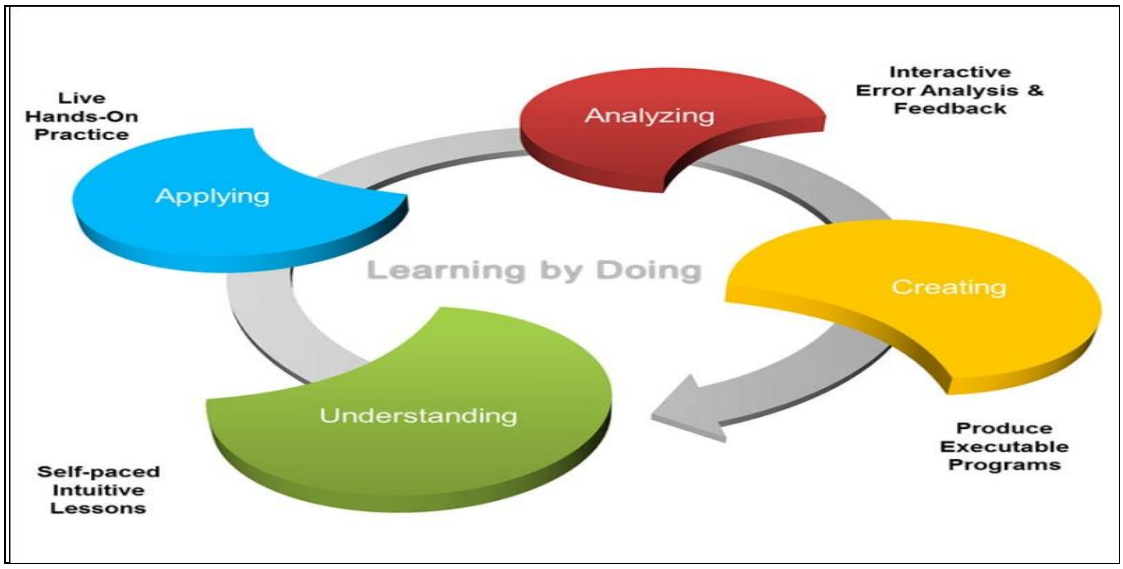
5CS53

II B.Tech II Semester

B.Tech. FOUR YEAR DEGREE COURSE

VNR Vignana Jyothi Institute of Engineering & Technology

Bachupally, Nizampet (S.O.), Hyderabad – 90



A great way of doing something is to do it

--By doing

And How do u learn to do something ?

By doing it with some real life examples.

As a part of Active teaching aid followed at VNRVJIET, This is a sample copy for the course "Java Programming ",the course which is being carried over for B.Tech, IInd year CSE students.

As a part of curriculum and syllabi, The following document shows few of the topics that have been covered with this learning by doing methodology.(ofcourse the other topics cover other teaching aids).Most of the exercises and topics have been taken up in lab and the topics have been made well understood to the students, in such a way that they were able to analyse the content and apply them accordingly in the lab. Also We have Course based projects concept that enable students to carry over the knowledge acquisition and apply it by bringing out the new ideas and developing a new concept or implementing new exercise. Students

had hands on experience with this methodology and could bring out their talents come true.

Also, the students can learn and have good exposure over subject by using the Spoken Tutorials.

The link has been displayed here, www.spoken-tutorial.org"
and can be downloaded as zip file by using the following path:
http://process.spoken-tutorial.org/images/1/1b/Download-Tutorials.pdf.

We do have Course base projects concept, in which students will be developing few of the projects for each of the laboratory concerned subject.

Some of the stuff has been documented as pictures and notes.

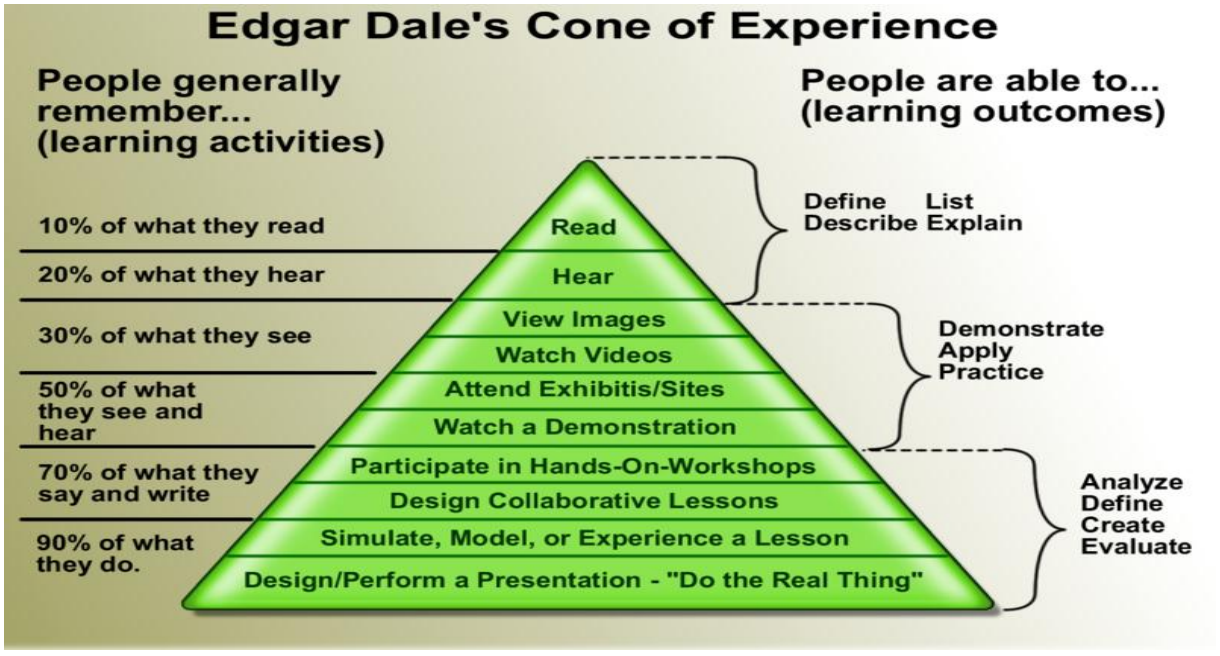
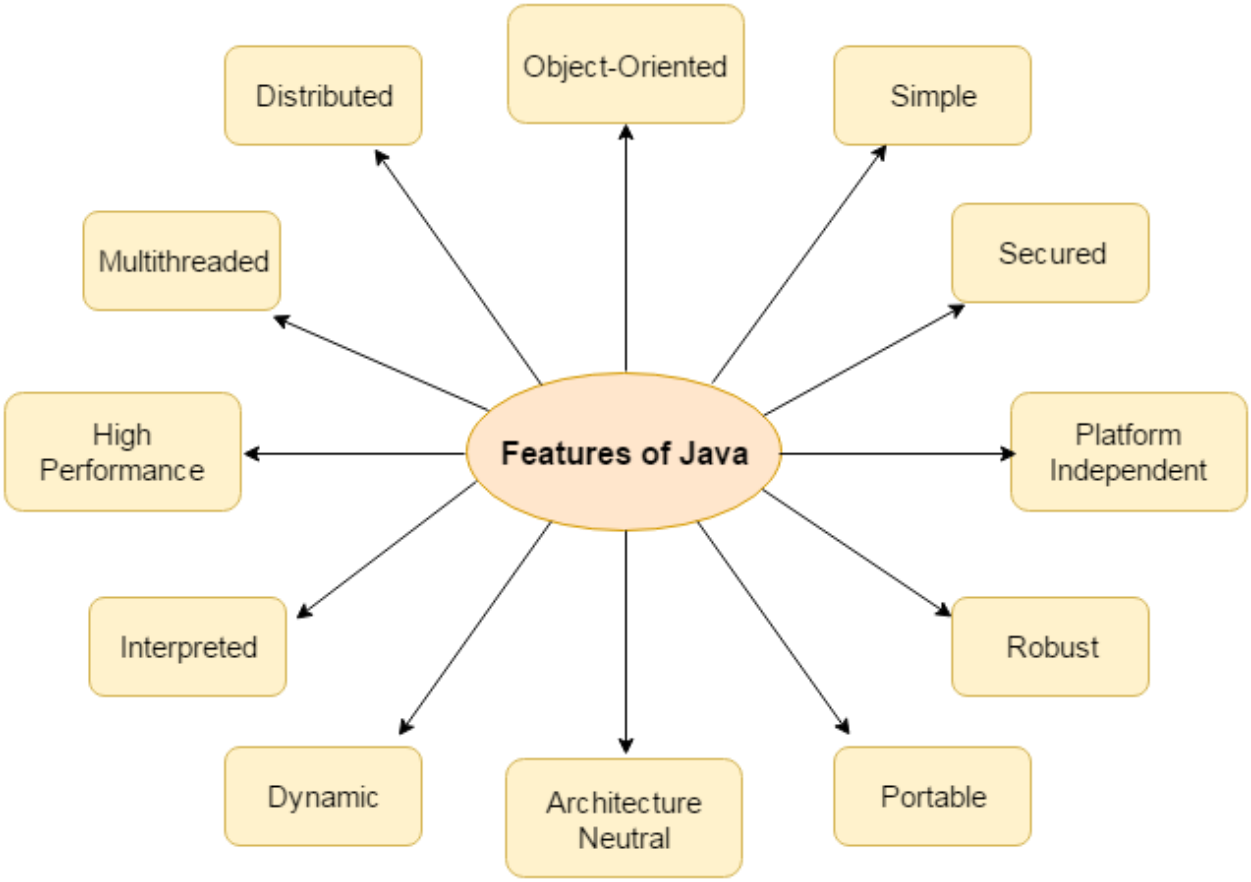
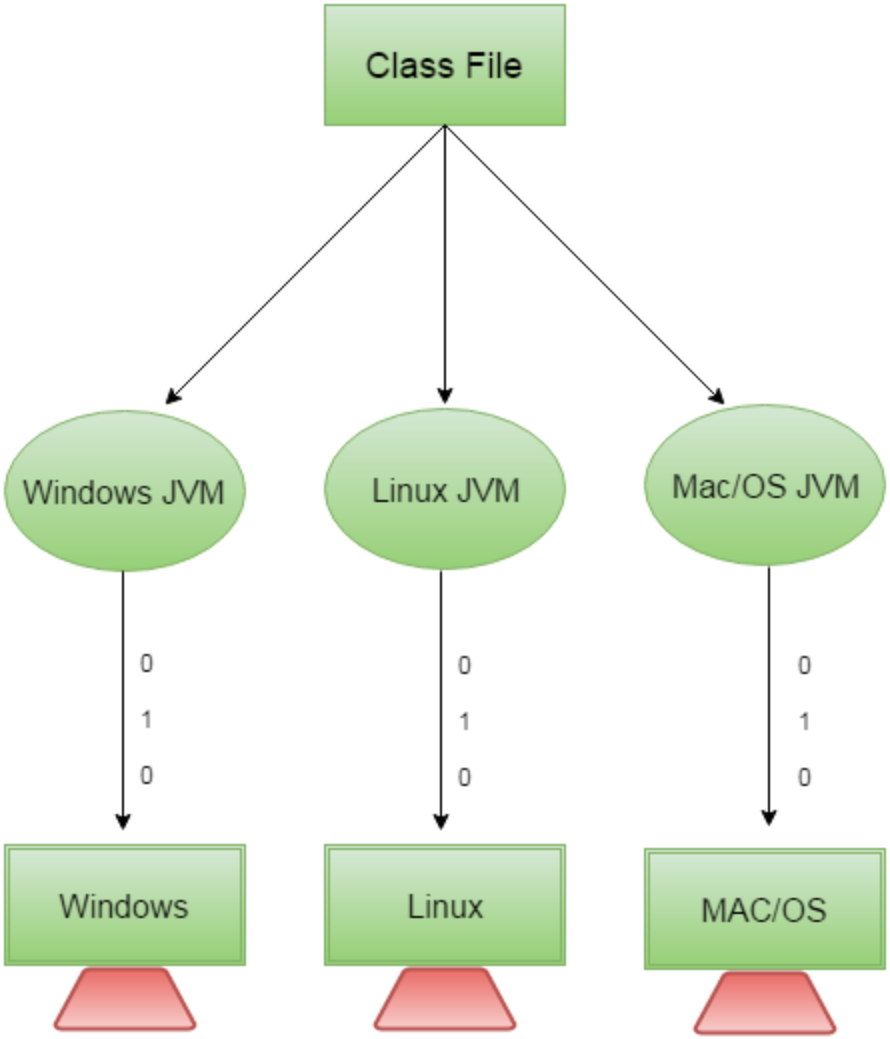


Figure1:The cone of learning pyramid

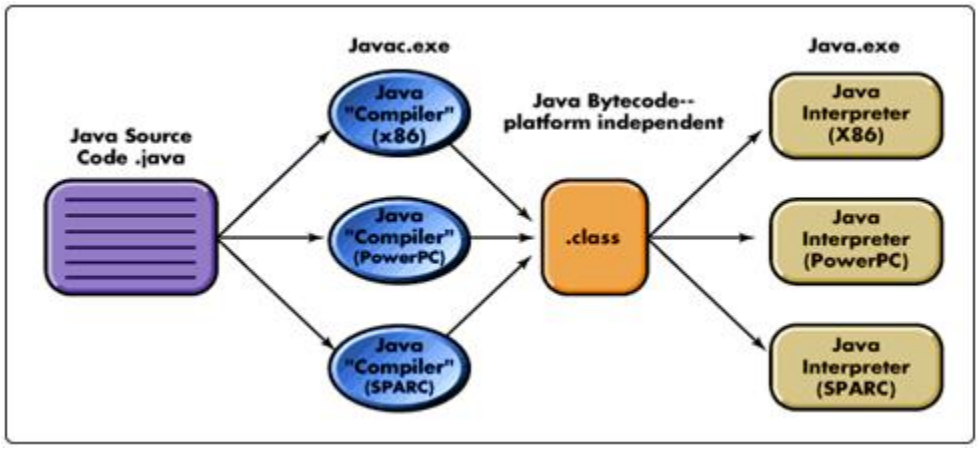
Activity no	Topic
1.	Features of Java.
2.	Is main method compulsory
3.	How does equals work on strings in Java
4.	Objects and classes
5.	Object Oriented concepts
6.	Polymorphism in Brief
7.	Inheritance and its types
8.	Why multiple inheritance is not supported in java?
9.	Exception handling in Java
10.	Multithread programming in Java
11.	Clue by sample snapshots
12.	Clues by sample input and output
13.	Course based projects
	References

Activity 1: Below are the features of Java.





Activity 2: Compilation of sample source code in Java



Activity 3: Is main method compulsory in Java?

The answer to this question depends on version of java you are using. Prior to JDK 5, main method was not mandatory in a java program.

- You could write your full code under static block and it ran normally.
  - The static block is first executed as soon as the class is loaded before the main(); method is invoked and therefore before the main() is called. main is usually declared as static method and hence Java doesn't need an object to call main method.

However, From JDK6 main method is mandatory. If your program doesn't contain main method, then you will get a run-time error "main method not found in the class". Note that your program will successfully compile in this case, but at run-time, it will throw error.

```
// This program will successfully run
// prior to JDK 5
public class Test
{
    // static block
    static
    {
        System.out.println("program is running without main() method");
    }
}
```

Output:

- If run prior to JDK 5

```
program is running without main() method
```

- If run on JDK 6,7,8

```
Error: Main method not found in class Test, please define the main method as:
   public static void main(String[] args)
Students run the above code ,and tested in lab and found the above resultsa
dn analysed the importance of main method in Java.
```

Activity 4: How does the equals work on Strings in Java

For this the following were considered including String and StringBuffer objects.

Consider the following codes in java:

```
// This program prints false

class GFG {


  public static void main(String[] args) {

    StringBuffer sb1 = new StringBuffer("GFG");

    StringBuffer sb2 = new StringBuffer("GFG");

    System.out.println(sb1.equals(sb2));

  }

}
```

Output: Students gave a try out in Lab and is being documented as follows.

```
false
// This program prints true

class GFG {


  public static void main(String[] args) {

    String s1 = "GFG";

    String s2 = "GFG";

    System.out.println(s1.equals(s2));

  }

}
```

Output:

```
true
```

The output is false for the first example and true for the second example. In second example, parameter to equals() belongs String class, while in second example it to StringBuffer class. When an object of String is passed, the strings are compared. But when object of StringBuffer is passed references are compared because StringBuffer does not override equals method of Object class.

For example, following first program prints false and second prints true.

```
// This program prints false

class GFG {



  public static void main(String[] args) {

    String s1 = "GFG";

    StringBuffer sb1 = new StringBuffer("GFG");

    System.out.println(s1.equals(sb1));

  }

}
```

Output:

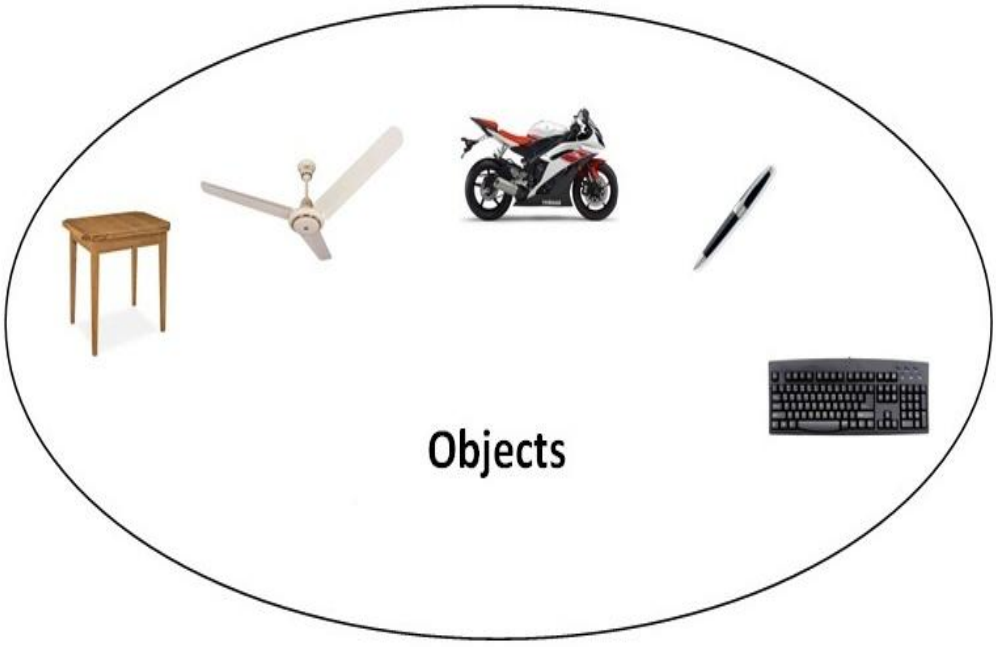
```
false
// This program prints true

class GFG {



  public static void main(String[] args) {

    String s1 = "GFG";

    StringBuffer sb1 = new StringBuffer("GFG");

    String s2 = sb1.toString();

    System.out.println(s1.equals(s2));

  }

}
```

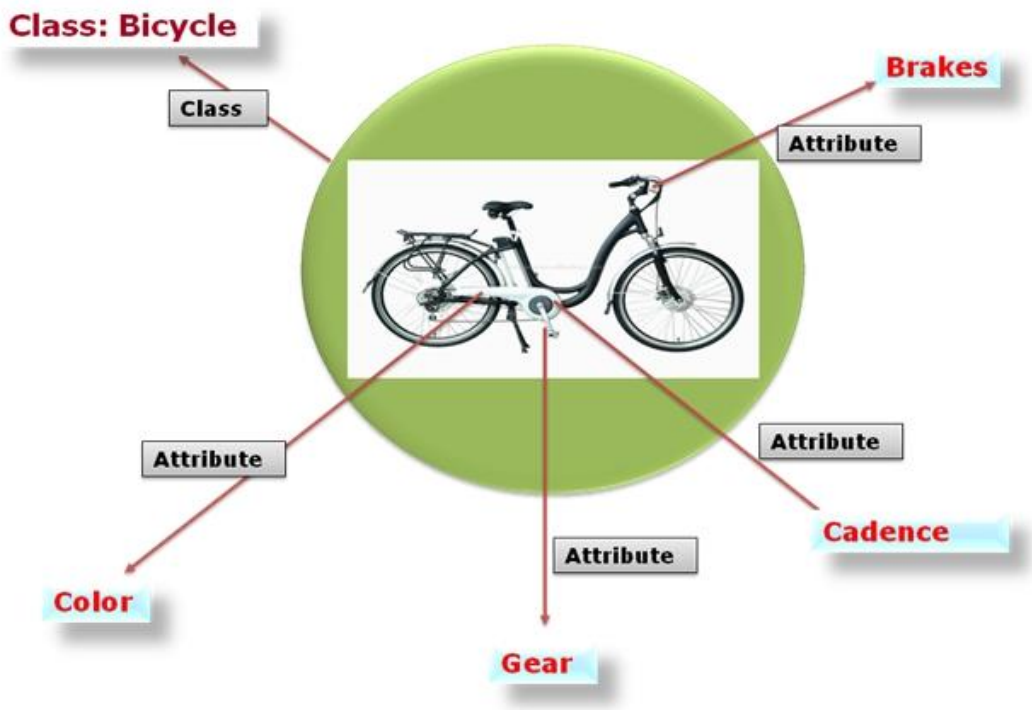
Output:

```
true
```

Activity 5: Objects and classes

Folowing are the images that were displayed in classroom , for students to identify them and ,response was very good, Students were able to identify them and Teacher has shown many such examples ,that made the class interesting and ever last learning basis for their programming skills.





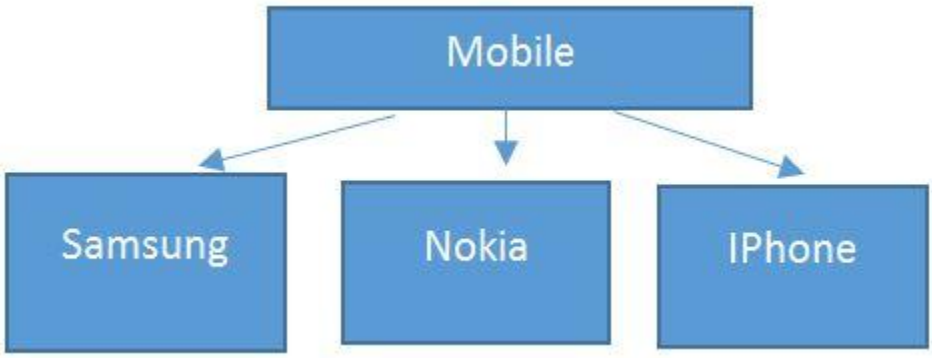
Object means a real word entity such as pen, chair, table etc.

Activity 6: Object Oriented Concepts in brief

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

A case Study of Mobile is taken to explain about the concept of objects and various types of inheritance.



In above diagram, each brand (Samsung, Nokia, IPhone) have their own list of features along with basic functionality of dialing, receiving a call & messaging.

When we talk about OOP, as the word indicate it will talk about an object (a real world object)

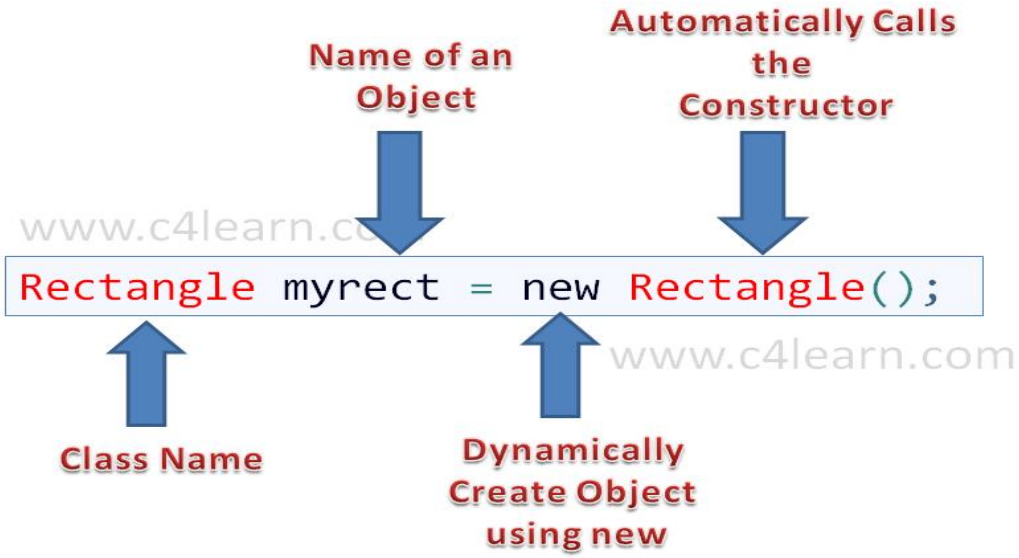
Objects

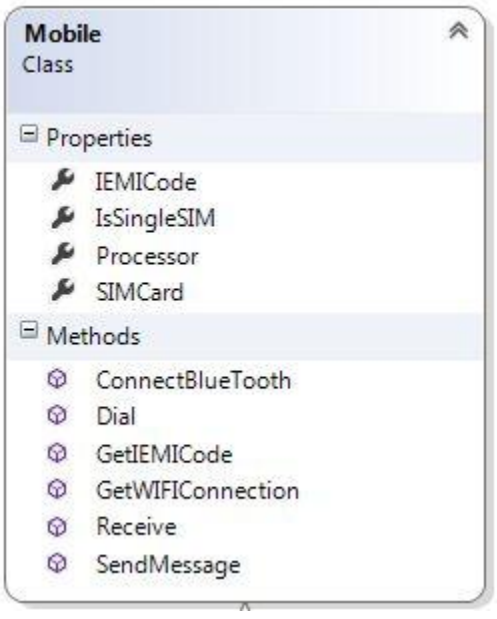
Any real world entity which can have some characteristics or which can perform some work is called as Object. This object is also called as an instance i.e. - a copy of entity in programming language. If we consider the above example, a mobile manufacturing company, at a time manufactures lacs of pieces of each model which are actually an instance. This objects are differentiated from each other via some identity or its characteristics. This characteristics is given some unique name.

1. Mobile mbl1 = new Mobile ();
2. Mobile mbl2 = new Mobile ();

Class

A Class is a plan which describes the object. We call it as a blue print of how the object should be represented. Mainly a class would consist of a name, attributes & operations. Considering the above example, A Mobile can be a class which has some attributes like Profile Type, IMEI Number, Processor, and some more.) & operations like Dial, Receive & SendMessage.





Encapsulation

Encapsulation:

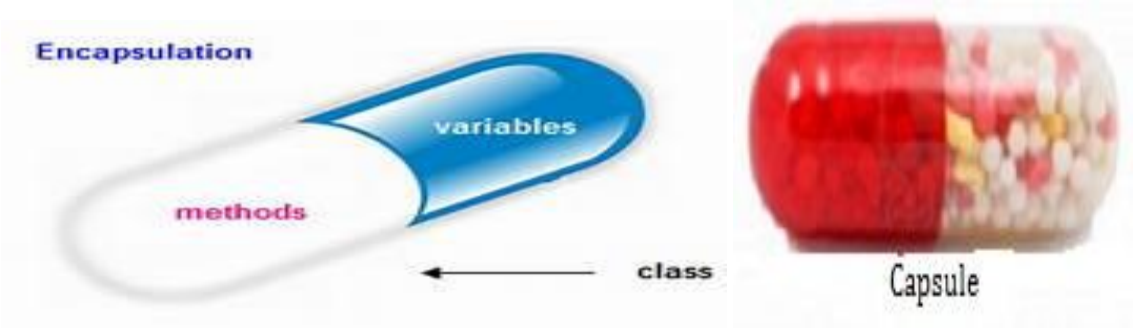
For the mathematical equation, shown in the figure assume that complex functions are required - >But I the end we obtain result for it

Calculator shows the result of equation but hides the implementation (calculating the result) involved.

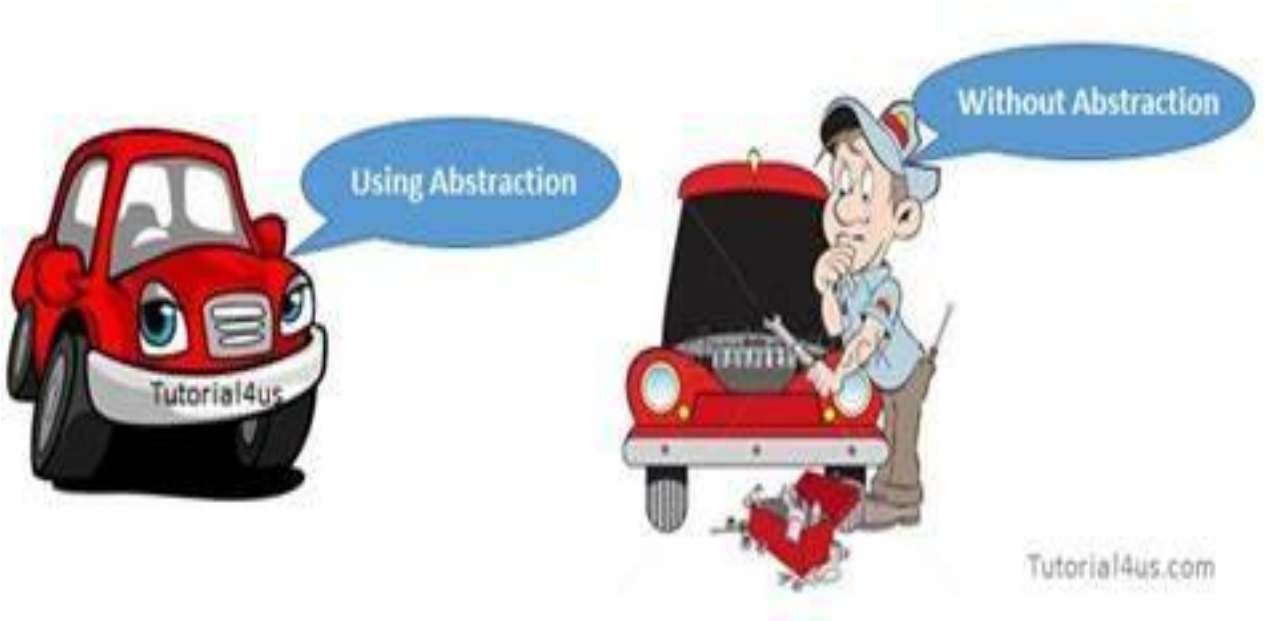
Abstraction:

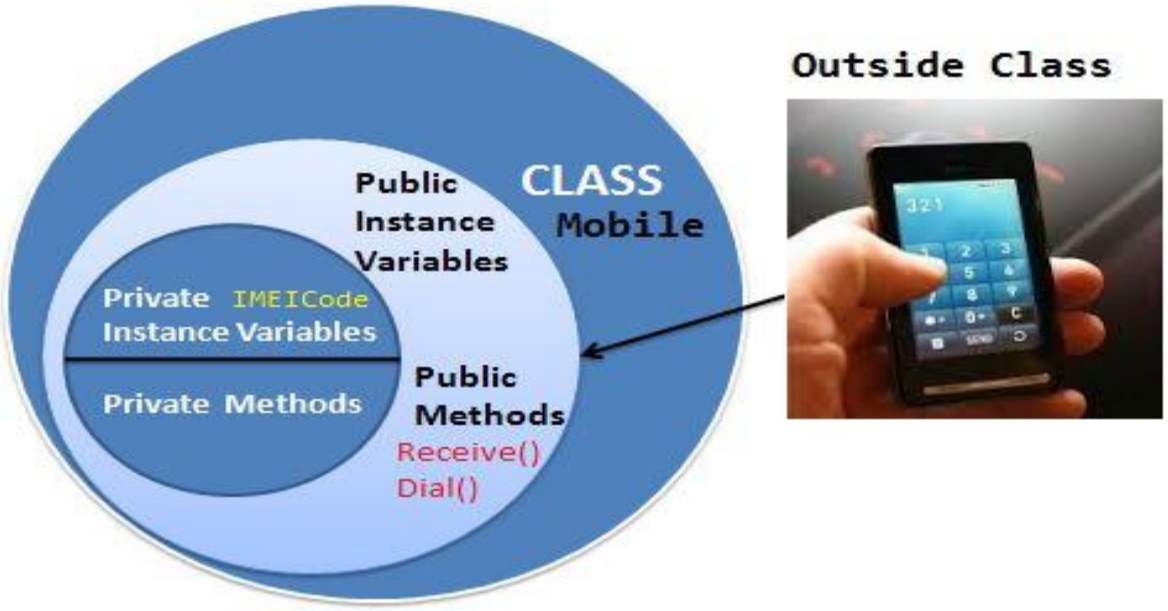
The calculator shown in the figure has to be powered by a battery source. How the battery module works for the calculator is not necessary to know for the user who uses the calculator.

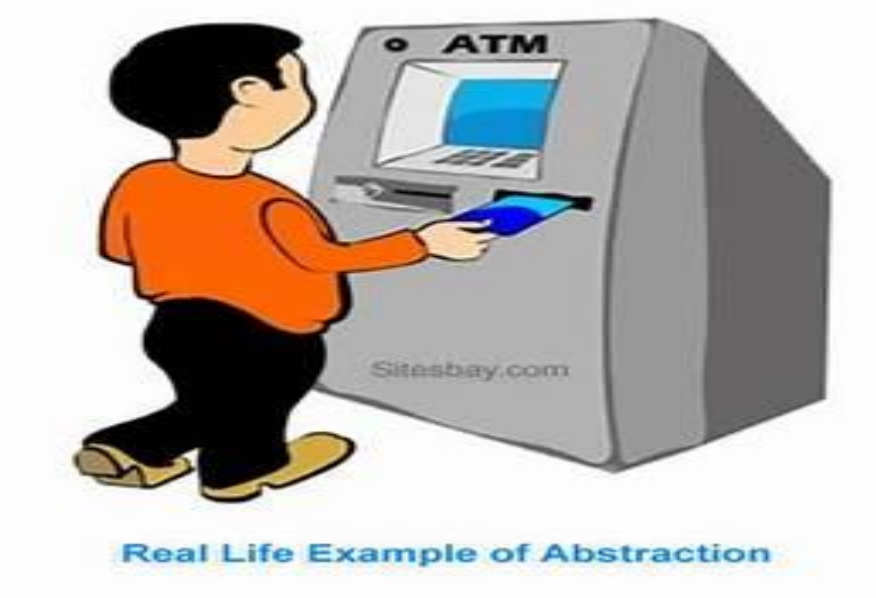
Using Battery module along with other modules we use calculator -> Thus using Abstraction encapsulation is performed



Abstraction:







Activity 7: Polymorphism as a whole

Polymorphism

In Shopping malls behave like Customer

In Bus behave like Passenger

In School behave like Student

At Home behave like Son Tutorial4us.com

State :

Phones with Different Colors

Phones of Different type and model no.

Behavior :

Make and Receive Calls

Send Messages

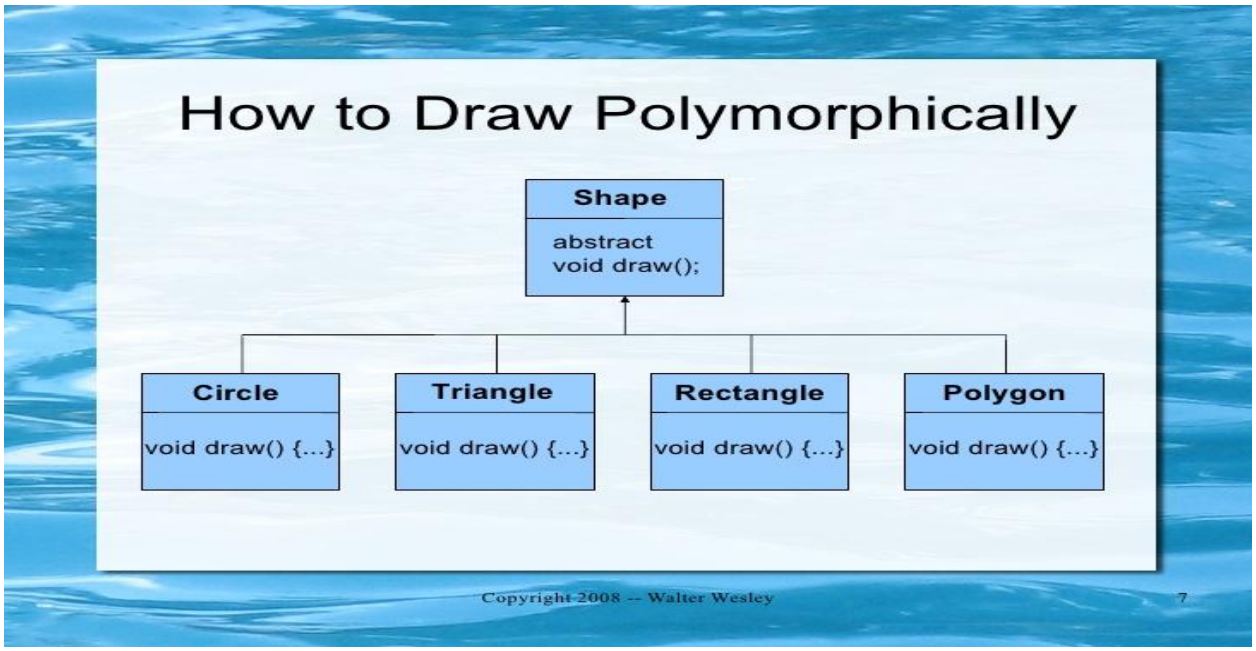
Set Alarms

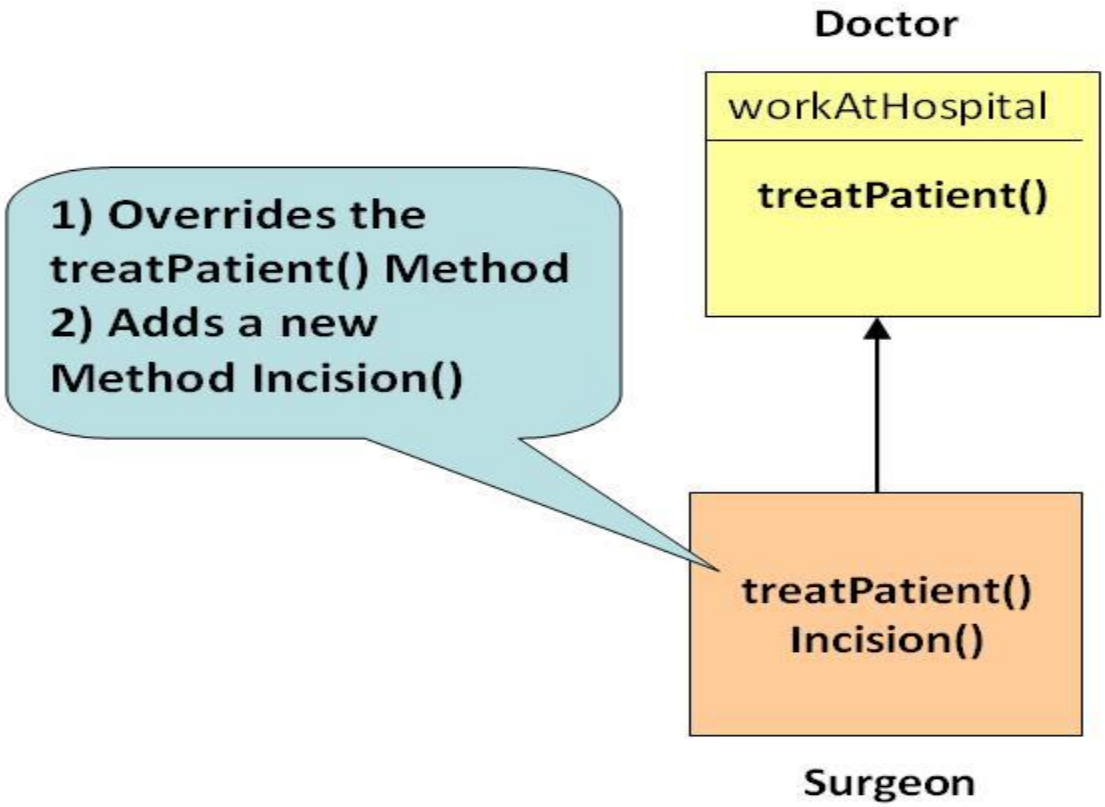
sharingwithcaring.com

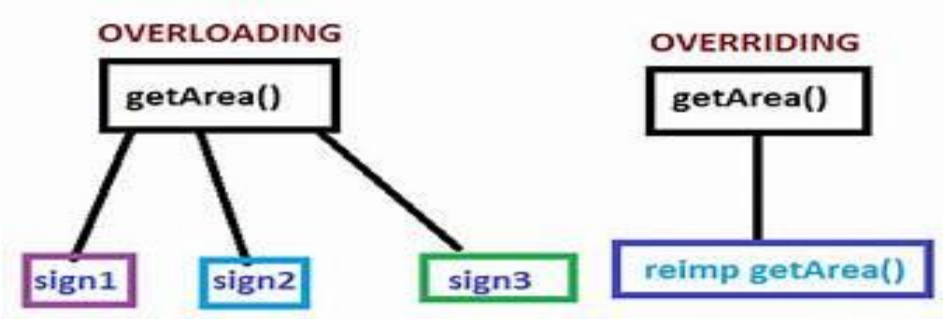
Ploymorphism

Method Overloading:
having multiple methods with same name but with different signature (number, type and order of parameters).

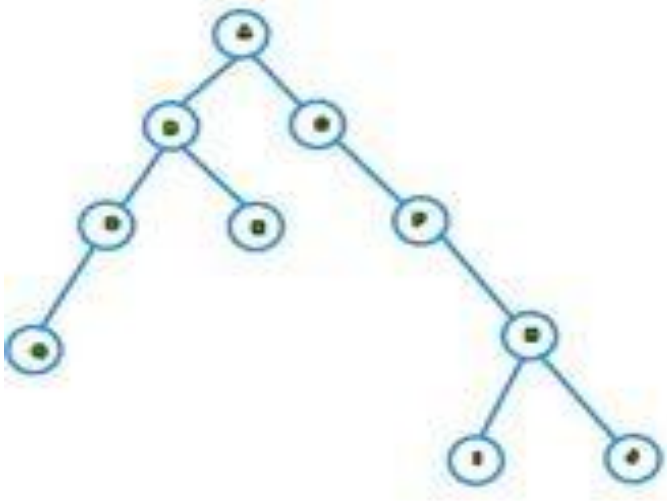
Method Overriding:
When a subclass contains a method with the same name and signature as in the super class then it is called as method overriding.


How to Draw Polymorphically
Shape
abstract
void draw();
Circle
void draw() {...}
Triangle
void draw() {...}
Rectangle
void draw() {...}
Polygon
void draw() {...}
Copyright 2008 -- Walter Wesley
7


Doctor
workAtHospital
treatPatient()
1) Overrides the treatPatient() Method
2) Adds a new Method Incision()
treatPatient()
Incision()
Surgeon



Activity 8: Inheritance and its types



Ability to extend the functionality from base entity in new entity belonging to same group. This will help us to reuse the functionality which is defined before.

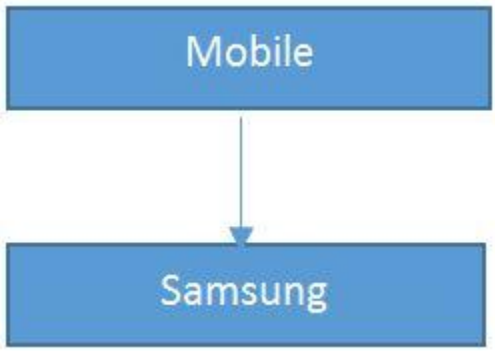
Considering the example, the above figure 1.1 itself shows what is inheritance. Basic Mobile functionality is to Send Message, dial & receive call. So the brands of mobile is using this basic functionality by extending the mobile class functionality and adding their own new features to their respective brand.

There are mainly 4 types of inheritance:

1. Single level inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Hybrid inheritance

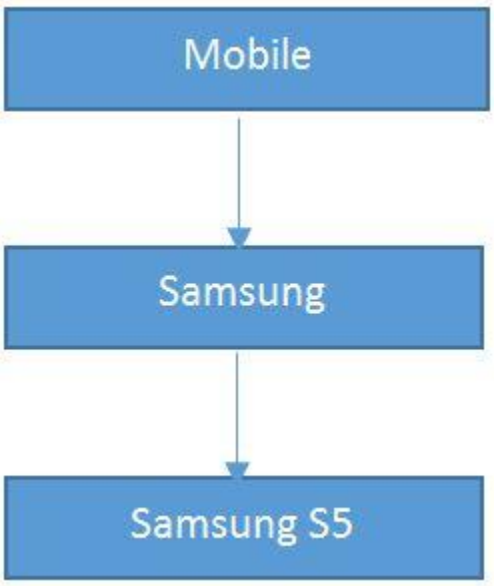
Single level inheritance

In Single level inheritance, there is single base class & a single derived class i.e. - A base mobile features is extended by Samsung brand.



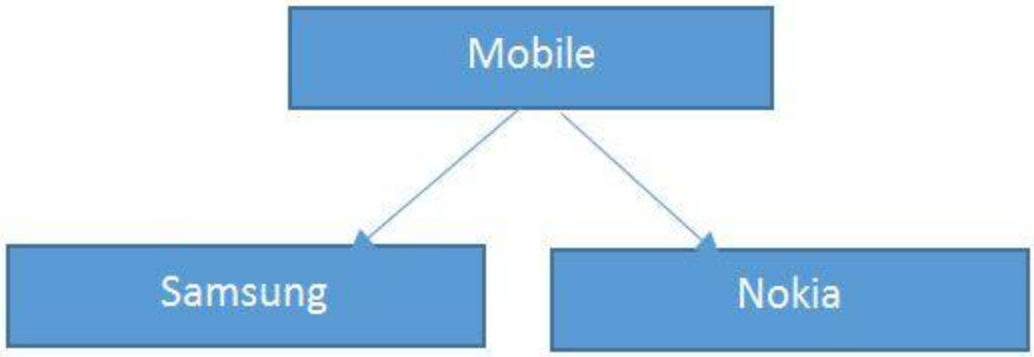
Multilevel inheritance

In Multilevel inheritance, there is more than one single level of derivation. i.e. - After base features are extended by Samsung brand. Now Samsung brand has manufactured its new model with new added features or advanced OS like Android OS, v4.4.2 (kitkat). From generalization, getting into more specification.



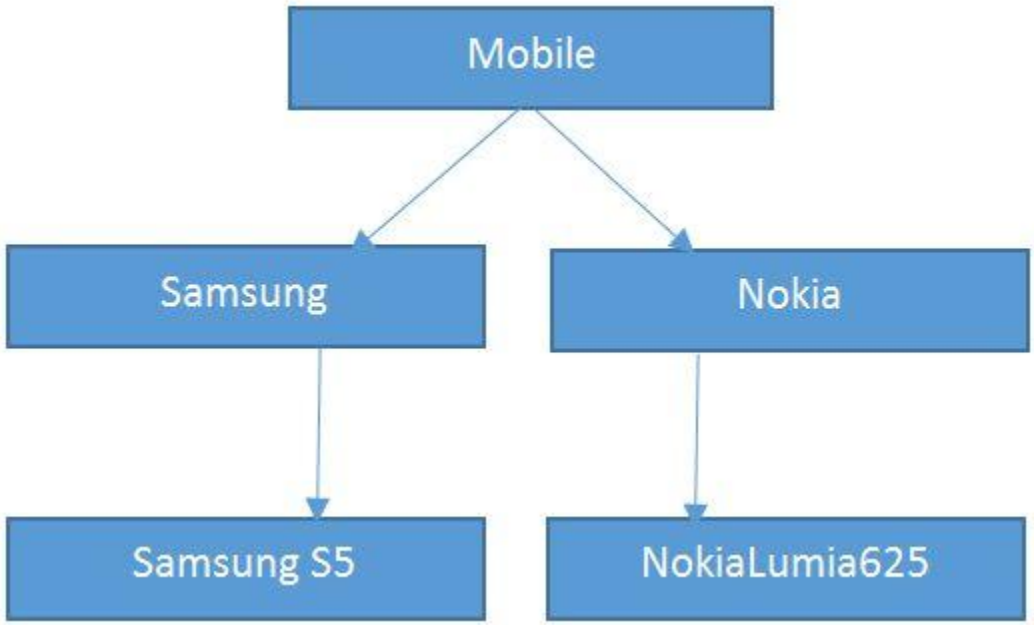
Hierarchal inheritance

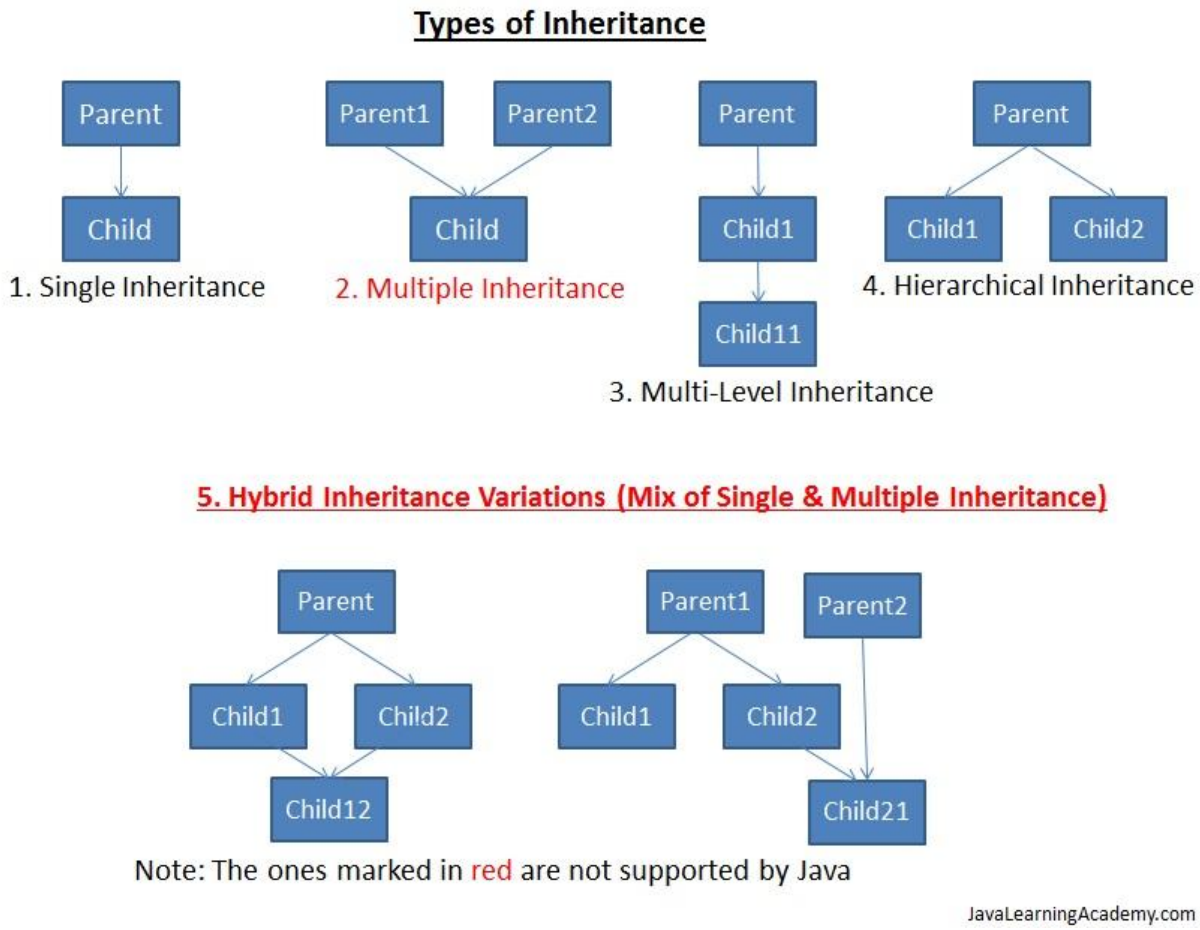
In this type of inheritance, multiple derived class would be extended from base class, it's similar to single level inheritance but this time along with Samsung, Nokia is also taking part in inheritance.



Hybrid inheritance

Single, Multilevel, & hierarchal inheritance all together construct a hybrid inheritance.





Activity 9: Why multiple inheritance is not supported in Java

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B and C are three classes. The C class inherits A and B classes. If A and B classes have same method and you call it from child class object, there will be ambiguity to call method of A or B class.

Since compile time errors are better than runtime errors, java renders compile time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error now.

```
1. class A{
2. void msg(){System.out.println("Hello");}
3. }
4. class B{
5. void msg(){System.out.println("Welcome");}
6. }
```

```
7.  class C extends A,B{//suppose if it were
8.
9.   Public Static void main(String args[]){
10.    C obj=new C();
11.    obj.msg();//Now which msg() method would be invoked?
12. }
13. }
14. Output:
```

15. **Compile by: javac C.java**

```
16. .212.213.229/C.java:7: error: '{' expected
    class C extends A,B{//suppose if it were
    ^
    .212.213.229/C.java:9: error: ';' expected
    Public Static void main(String args[]){
    ^
    2 errors
```

Interface

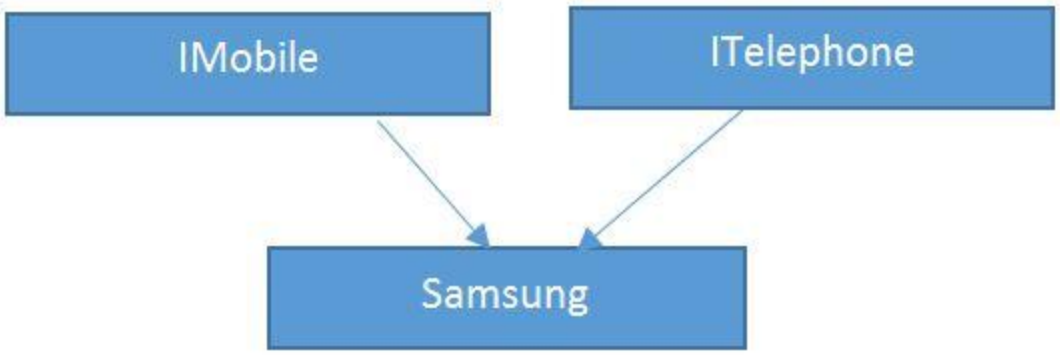
Multiple inheritance where derived class will extend from multiple base classes.

Samsung will use the function of multiple Phone (Mobile & Telephone). This would create a confusion for complier to understand which function to be called when any event in mobile is triggered like Dial () where Dial is available in both the Phone i.e. - (Mobile & Telephone). To avoid this confusion C# came with the concept of interface which is different from multiple inheritance actually.

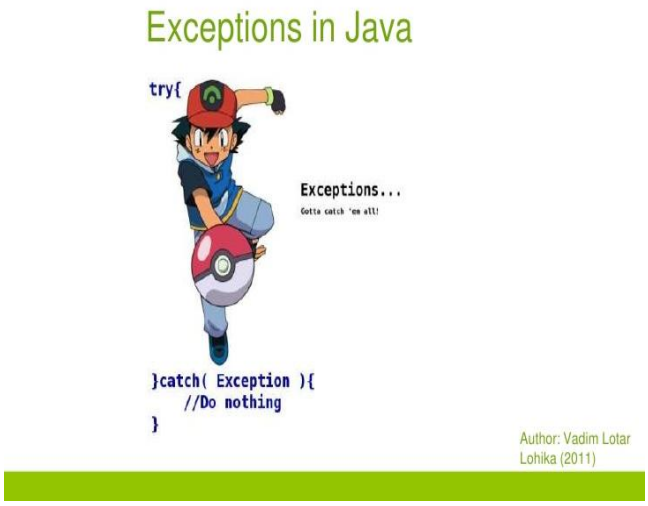
If we take an interface it is similar to a class but without implementation & only declaration of properties, methods, delegates & events. Interface actually enforces the class to have a standard contract to provide all implementation to the interface members. Then what's is the use of interface when they do not have any implementation? Answer is, they are helpful for having readymade contracts, only we need to implement functionality over this contract.

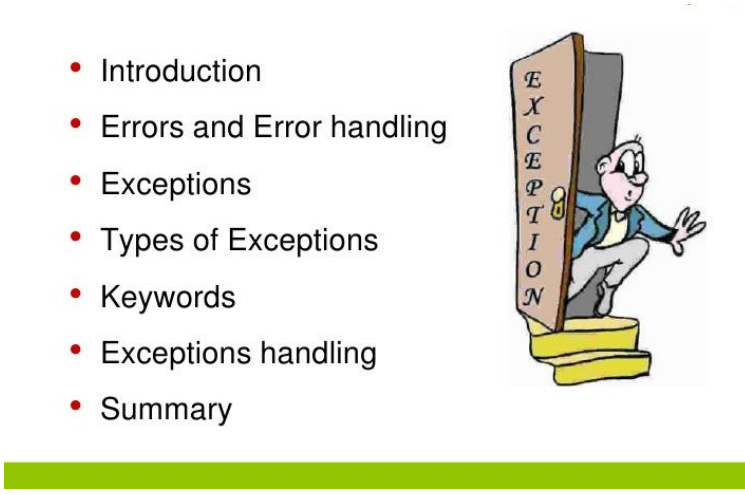
I mean to say, Dial would remain Dial in case of Mobile or Telephone. It won't be fair if we give different name when its task is to Call the person.

Interface is defined with the keyword 'interface' .All properties & methods with in the interface should be implemented if it is been used. That's the rule of interface.

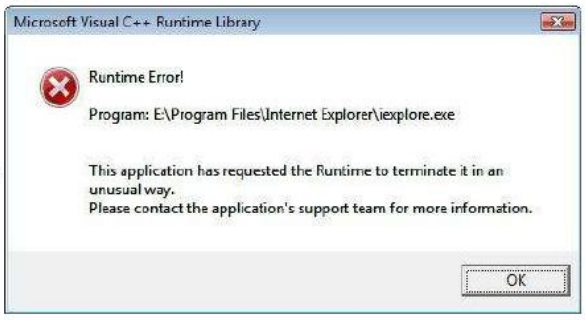


Activity 9: Exception handling in Java



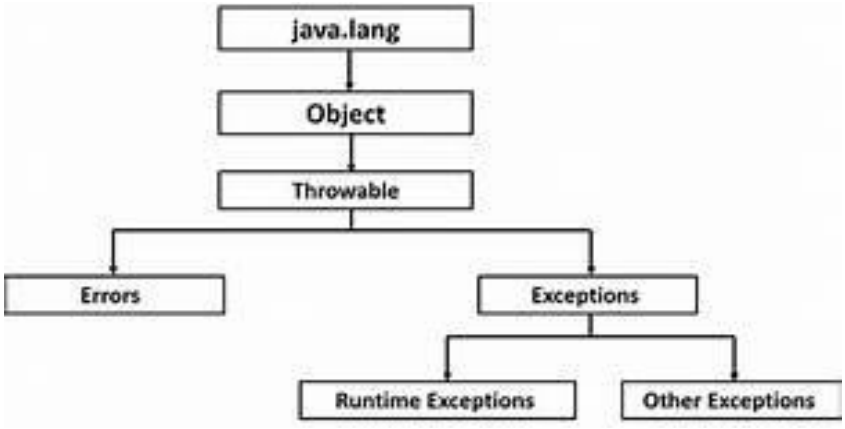


- Unhandled errors may manifest themselves as incorrect results or behavior, or as abnormal program termination.
- Errors should be handled by the programmer, to prevent them from reaching the user.



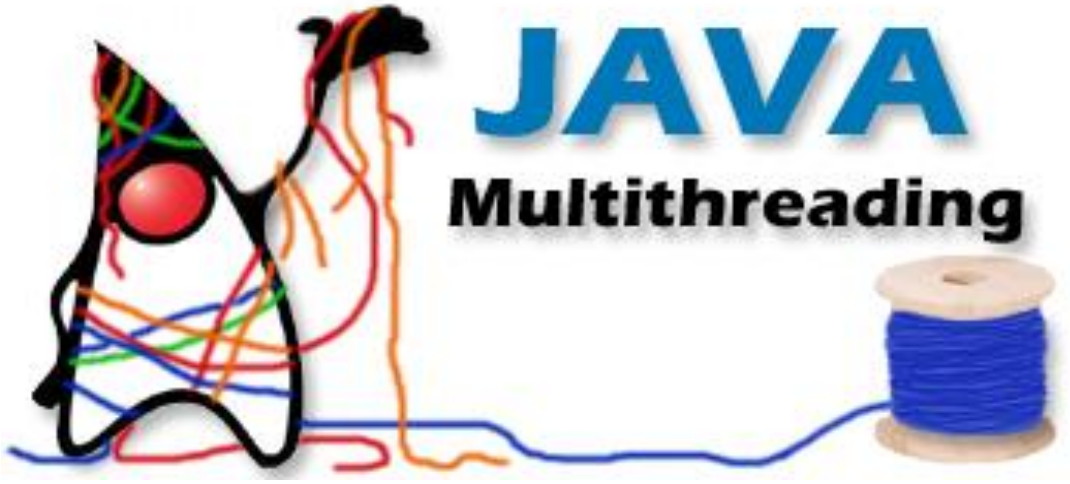
Exceptions

lóhika



Activity 10: multithreading in Java

A thread is like the electricity passing through the wire to the devices.
A single wire or Multi wire can power the devices to run.



Multitasking - Multithreading

Handling multiple tasks – Multitasking
Ability to initiate multiple processes – Multithreading

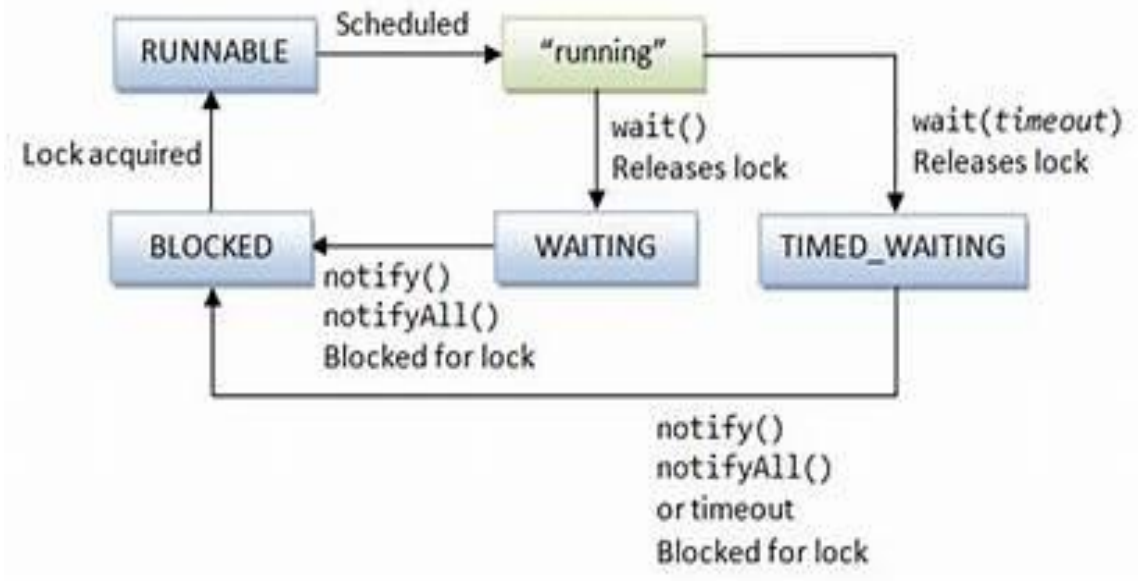
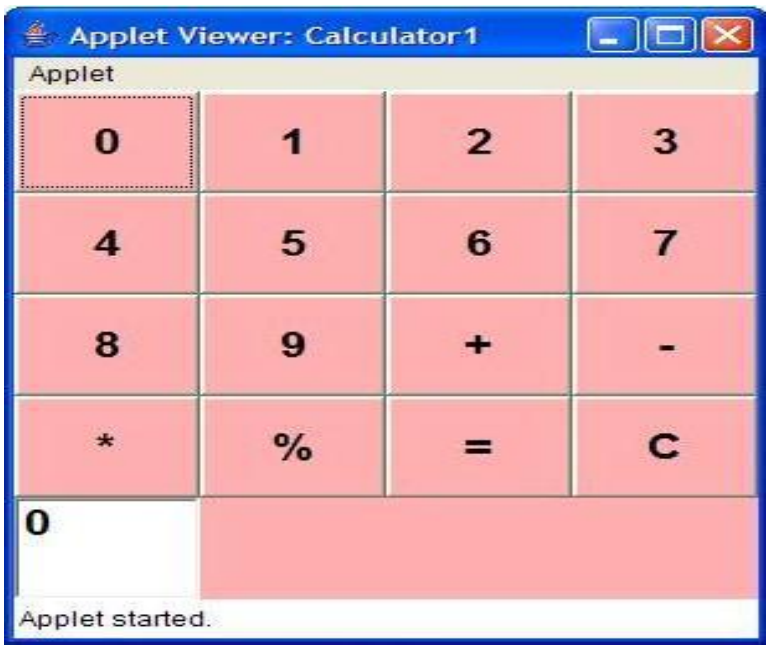
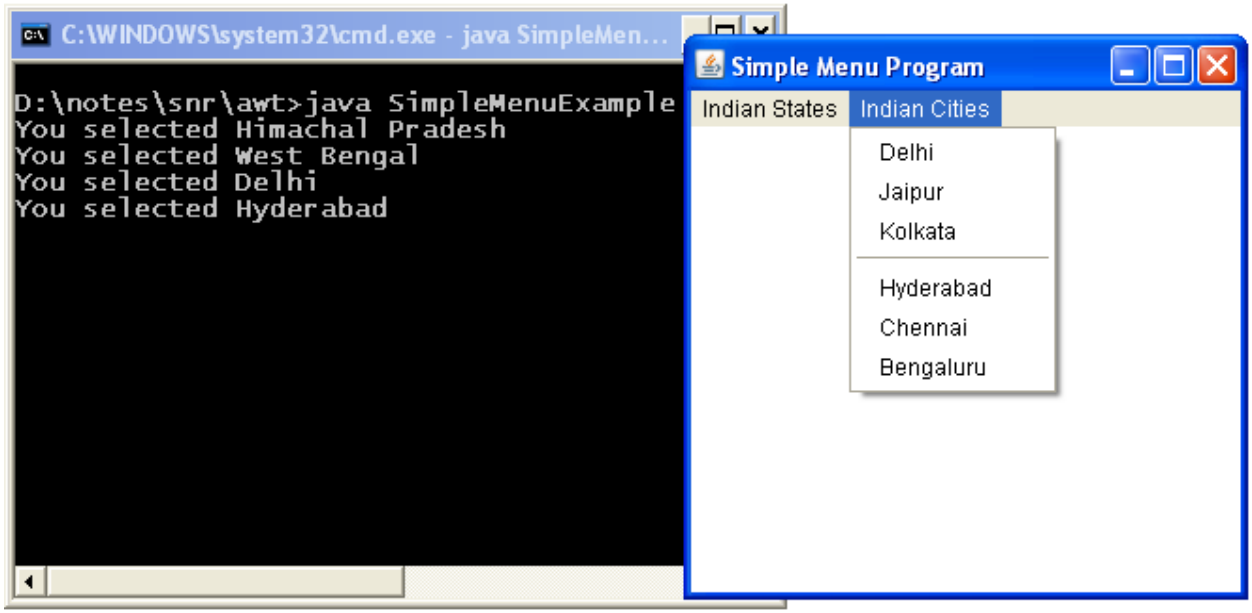


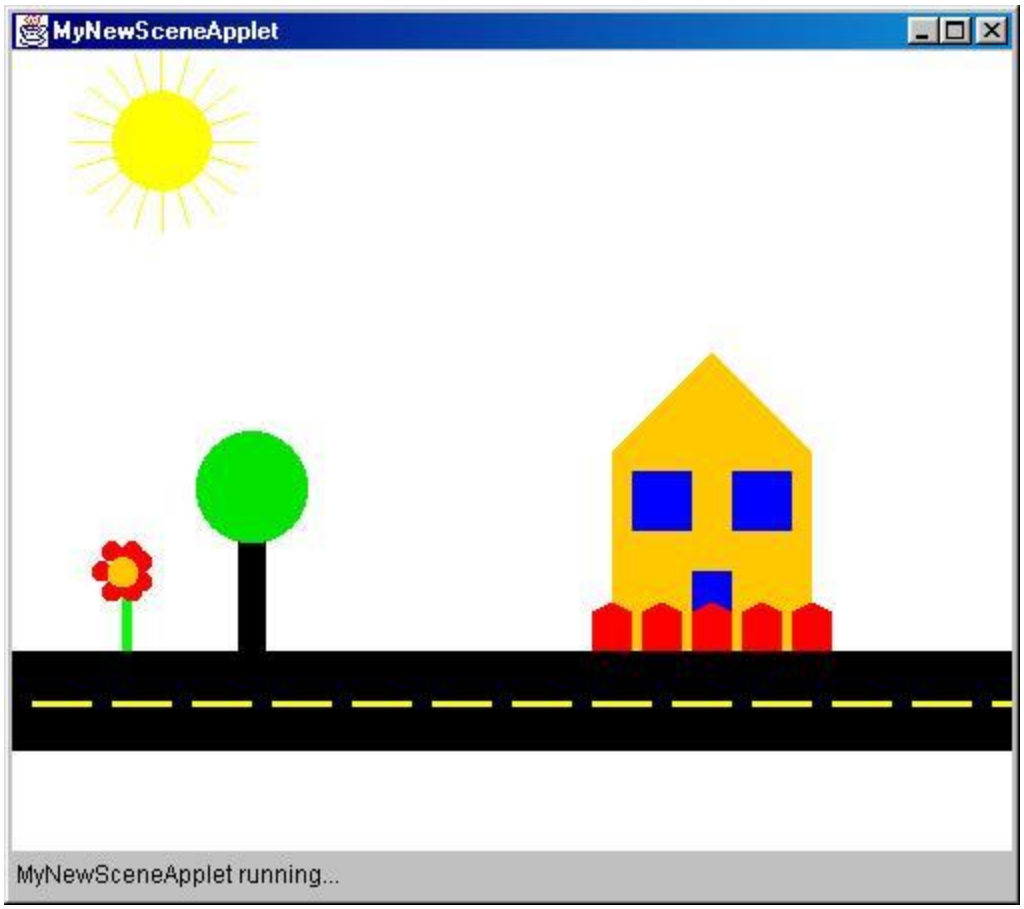
Figure: Life cycle of Thread

Activity 11: Applets in Java,Clue by sample snaphosts

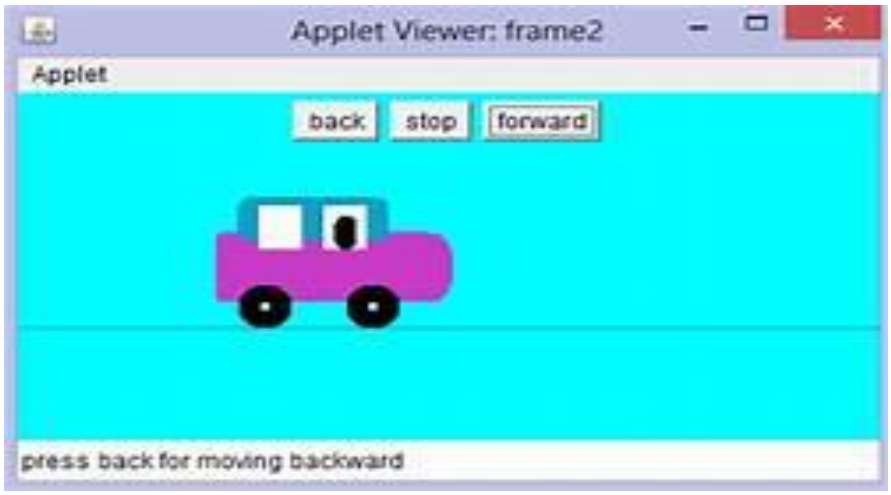
Sample snapshots of the layouts that need to be developed through program:


Applet Viewer: Calculator1
Applet
0
1
2
3
4
5
6
7
8
9
+
-
*
%
=
C
0
Applet started.


C:\WINDOWS\system32\cmd.exe - java SimpleMen...
D:\notes\snr\awt>java SimpleMenuExample
You selected Himachal Pradesh
You selected West Bengal
You selected Delhi
You selected Hyderabad
Simple Menu Program
Indian States
Indian Cities
Delhi
Jaipur
Kolkata
Hyderabad
Chennai
Bengaluru


MyNewSceneApplet
MyNewSceneApplet running...


Applet Viewer:...
Applet
JAVA
JAVA
Applet started.



.

Activity 12: Clues by sample input and output

1.)

Develop a code for the respective inputs and outputs.

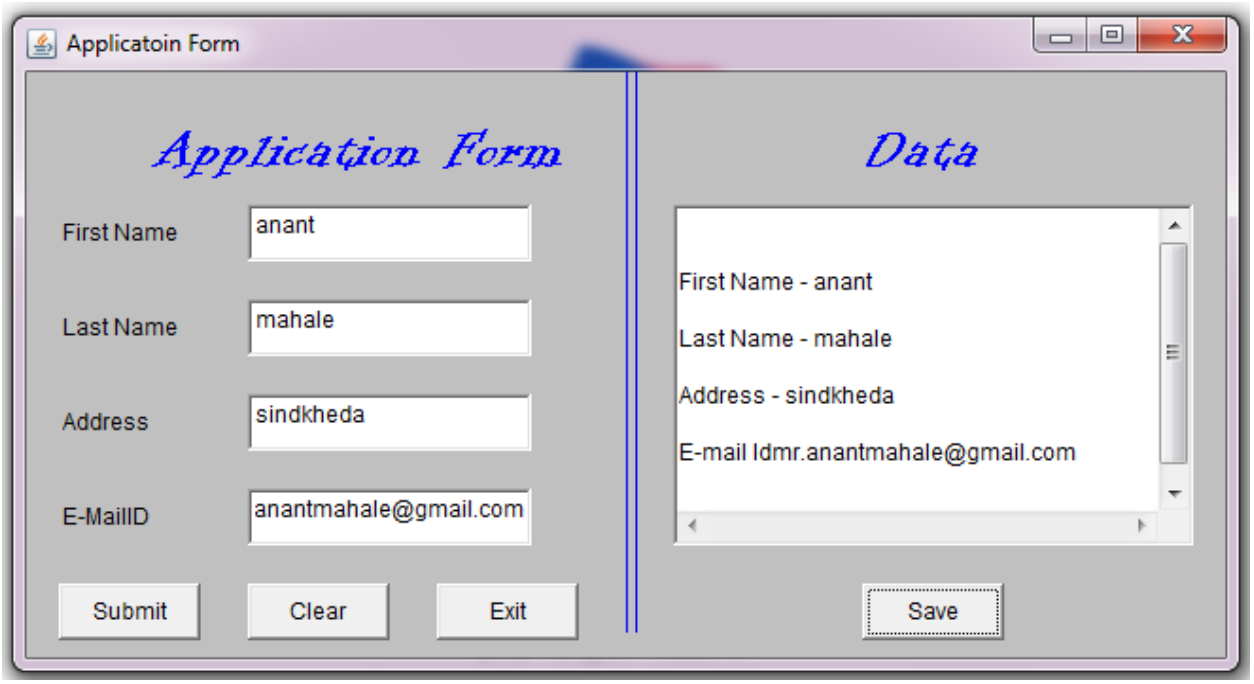
Input:
First line of the input contains an integer T which denotes the number of test cases. Then T test cases follow. Each test case contains two lines. First line contain 1 for circle or 2 for rectangle.Second line contains integers R(radius) or L(length) and B(Breadth)(space separated).
Output:
For each test case, print area of circle or rectangle.
Constraints:
1 <= T< = 10
1<=R<=100
1<=L<= 100
1<=B<= 100
Note: Use Math.PI for pi value.
Example:
Input:
2
1
5
2
5 10

Output:
78.53981633974483
50

2.) Develop a generalized for of this zigzag pattern:

n=4
1
3*2
4*5*6
10*9*8*7
10*9*8*7
4*5*6
3*2
1

3.)
Try to develop a sample application form



Activity 13: Course Based Projects :

1. Java Project on Traffic Jam Detection

2. Java Project on Student Information System

3. Java Project for Online Chat

4. Java Project for Login Authentication with Smart Card Network Security

5. Java Project for Content-Based Image Retrieval

6. Java Project Cheating Prevention in Visual Cryptography

7. Java Project for Voice Enable Smart Browser

8. Multi-Purpose Java Instant Messaging Tool

9. Java Project for Digital Image Processing

10. Intranet Mailing System

11. Java Network File Sharing System

12. Face Identification System

13. Text Editor in Java

14. Vehicle Identification System

References:

1. Standard Text Books like "Java Complete Reference" by Herbert Schildt any edition after 5th .
2. **"*Java in a Nutshell*" by Benjamin J. Evans & David Flanagan**
3. ***Introduction to Object-Oriented Programming with Java.*"**
   by C.Thomas wu
4. ***The Java Programming Language"* by Ken Arnold,James**
   Gosling,David Holmes.

For Beginner:

- Head First Java, 2nd Edition
- Thinking in Java (4th Edition)
- Think Java
- Introduction to Java by Sedgewick
- Java in a Nutshell
- Core Java Volume I--Fundamentals (9th Edition) (Core Series): Cay S. Horstmann
- Java How To Program (late objects) by Paul Deitel, Harvey Deitel

For Intermediate:

- Effective Java (2nd Edition): Joshua Bloch
- Java Performance: Charlie Hunt, Binu John
- Head First Servlets and JSP
- SCJP by Kathy and Sierra
- Java - The Complete Reference by Herbert Schildt.
- Java Concurrency in Practice
- Java Performance
- The Java Programming Language, 4th Edition

For Advanced:

1. Java Puzzlers : Traps, Pitfalls, And Corner Cases

Some of the urls

1. https://programmingbydoing.com/
2. https://www.learneroo.com/modules/11/nodes/269
3. https://www.codecademy.com/learn/learn-java/
4. https://www.robinosborne.co.uk/2015/05/13/learning-by-doing-java-maven-seyren-hipchat-github/