OOP & JAVA BASICS

FSMK

OOP

- Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are instances of classes, are used to interact with one another to design applications and computer programs
- Break down the huge problem into smaller entities or objects. Develop each objects separately. Define proper interface for each object. Put together all the objects to build application
- Example – Motor Car

  Different parts such us, wheels, doors, windows, engine are manufactured separately and get assembled to build a car.

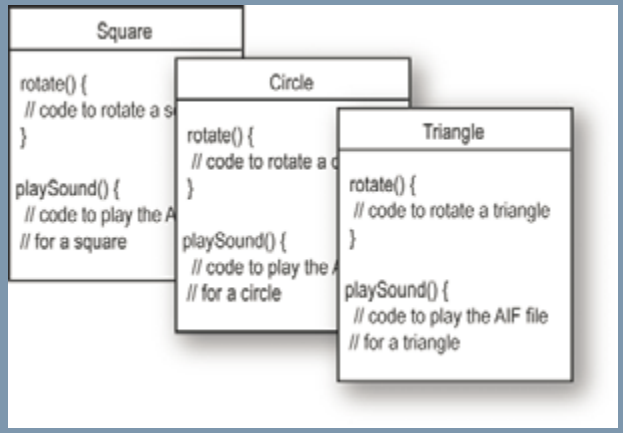
OOP

On click of shapes (square circle and triangle) rotate the shapes and play AIF file

Structured Programming

```
rotate(shapeNum) {
// make the shape rotate 360º
}
playSound(shapeNum) {
// use shapeNum to lookup which
// AIF sound to play, and play it
}
```

OOP



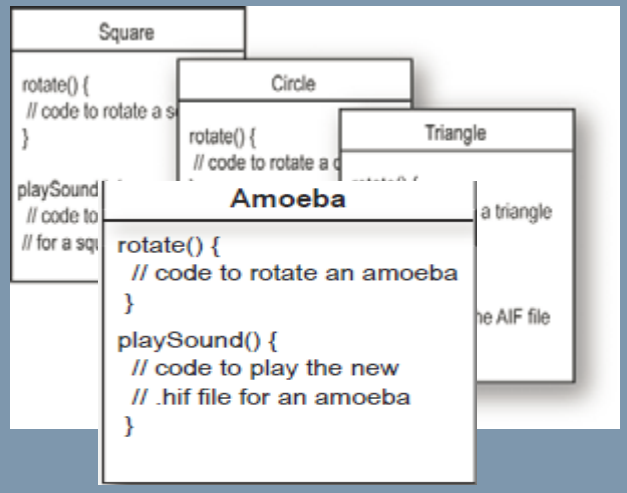
OOP

Include Amoeba shape which will rotate and play mp3 file

Structured Programming

```
rotate(shapeNum) {
// make the shape rotate 360º
}
playSound(shapeNum) {
// if the shape is not an amoeba,
// use shapeNum to lookup which
// AIF sound to play, and play it
// else
// play amoeba .mp3 sound}
```

OOP



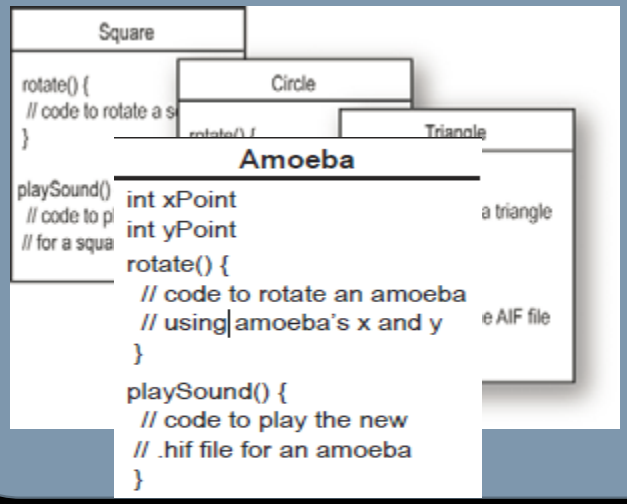
OOP

Only for Amoeba rotation should be based on x,y coordinate for others it should be center

Structured Programming

```
rotate(shapeNum, xPt, yPt) {
// if the shape is not an amoeba,
// calculate the center point
// based on a rectangle,
// then rotate
// else
// use the xPt and yPt as
// the rotation point offset
// and then rotate
}
```

OOP



JAVA

- Object Oriented Language
- Portable - Write once run anywhere
- Memory Management

How to Start

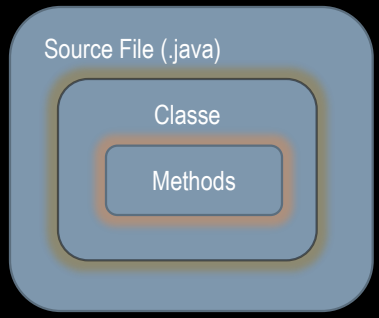
Install JDK

Sudo apt-get install openjdk-6-jdk

Write source code file (.java extension)

Compile it using javac compiler

Run the compiled bytecode on a JVM



SAMPLE JAVA CODE

```
public class MyFirstApp {
        public static void main(String[] args){
                System.out.println(“WoW!! You got me running");
        }
}
```

OOP – CLASSES AND OBJECTS

- A class defines the available characteristics and behaviour of a set of similar objects

  Class defines a type.

- Objects are created based upon class. Objects are concrete. It has state and behviour.

- Example – Vehicle class has methods steer(), brake() etc. And properties like color, number of doors.

Objects such us car, bike can be created based on Vehicle with its own properties.

Car	Racing Car	Tank	Tricycle
			
Colour: *Red* NumberOfDoors: *2* NumberOfWheels: *4* TopSpeed: *60mph*	Colour: *Yellow* NumberOfDoors: *0* NumberOfWheels: *4* TopSpeed: *200mph*	Colour: *Green* NumberOfDoors: *1* NumberOfWheels: *12* TopSpeed: *40mph*	Colour: *Orange* NumberOfDoors: *0* NumberOfWheels: *3* TopSpeed: *50mph*

JAVA CLASSES AND OBJECTS

- Class is a blue print by using this we create an object. It's a type specification

```
Public class Bank {
//member variables
//methods
}
```

- Objects have state and behavior.

```
Bank sbi = new Bank();
Sbi.membervariables
Sbi.methodss
```

Memory is allocated and a reference is created to the object

OOP – ENCAPSULATION

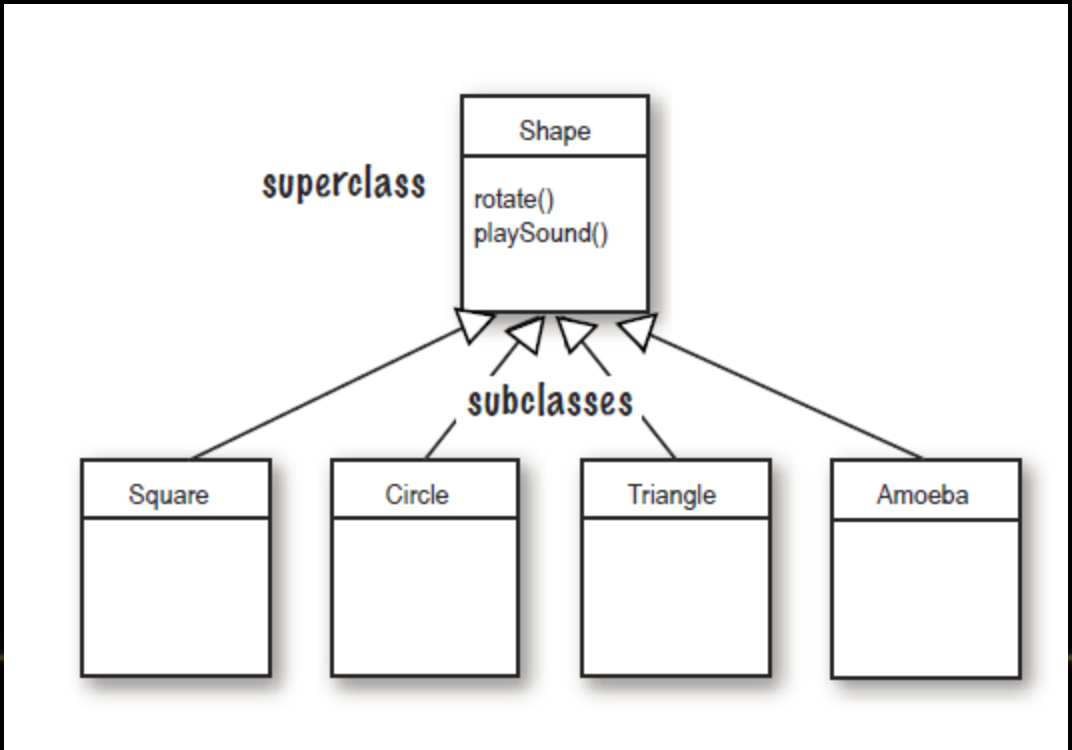
- A language mechanism for restricting access to some of the object's components. Hide private data from other classes. Access to the data is provided through public methods.

Java supports encapsulation by providing access modifiers.

public, private, protected and static member, variables and methods

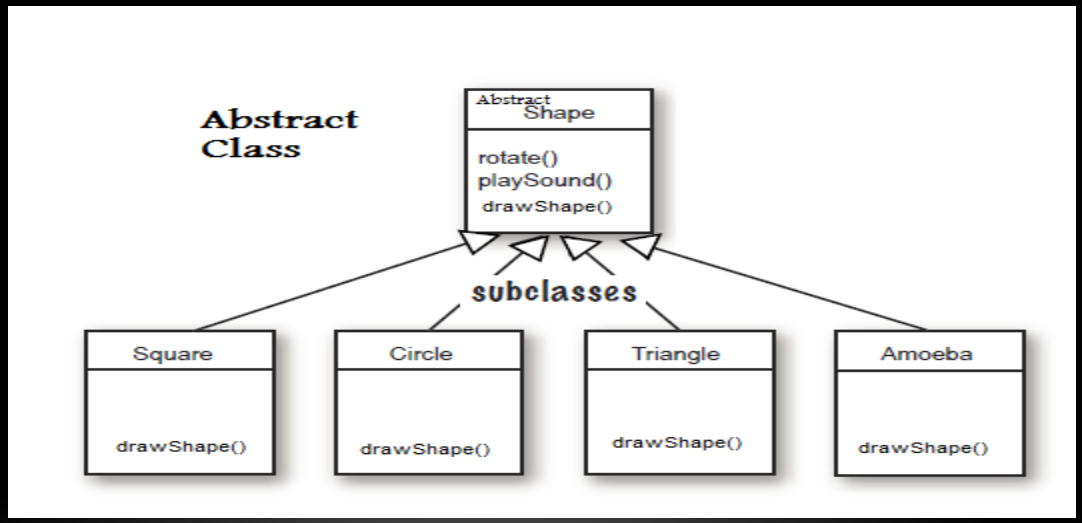
OOP - INHERITANCE

- For all the shapes, rotate and playsound methods are common. Move the common code to superclass. Other classes can extend the superclass. When a sub class extends a superclass, it inherits the member variables and methods of the superclass.



OOP - ABSTRACTION

- Abstraction in Object oriented programming is a way to segregate implementation from interface
- Abstraction in Java is achieved by using interface and abstract class in Java. In order to use interface or abstract class we need to extend and implement abstract method with concrete behavior. Interface or abstract classes cannot be instantiated.
- Interface in Java is 100% Abstract class.



INTERFACE

- Interfaces form a contract between the class and the outside world. This contract is enforced at build time by the compiler.

Interface definition –

```
interface Car {
void checkSpeed();
 void changeGear(int newValue);
}
```

class Swift **implements** Car{

```
 // remainder of this class // implemented as before
 }
```

OOP - POLYMORPHISM

- To the same message, at run time, the behaviour differs based on the exact type of object.
- Create reference of sub classes to super class. Binding between the actual object to the reference happens at runtime. This is called late binding.
- Exact behaviour is determined at runtime.

```
IShape shapeObj[] = new IShape[4];
shapeObj[0] = new Square();
shapeObj[1] = new Circle();
for(int i=0; i<shapeObj.length;i++){
shapeObj[i].draw();
shapeObj[i].rotate();
shapeObj[i].playSound();
}
```

JAVA INNER CLASS

- **Inner classes**, also called ***Nested Classes***, are nothing but classes that are defined within other classes.
- The class can be anonymous without any name. We can instantiate an anonymous inner class without actually making a separate class.
- Inner classes can be called from inside the code of a function. Only visible inside the function.

```
button.addActionListener(new ActionListener()
 {
 public void actionPerfored(ActionEvent e) {
// do something.
 }
});
```

JAVA EXCEPTION HANDLING

An exception is a problem that arises during the execution of a program.

- Try, catch and finally are used to catch an exception

```
try {
        //Protected code
 }catch(ExceptionName e1) {
        //Handle Exception
}finally{
        //clean up code
}
```

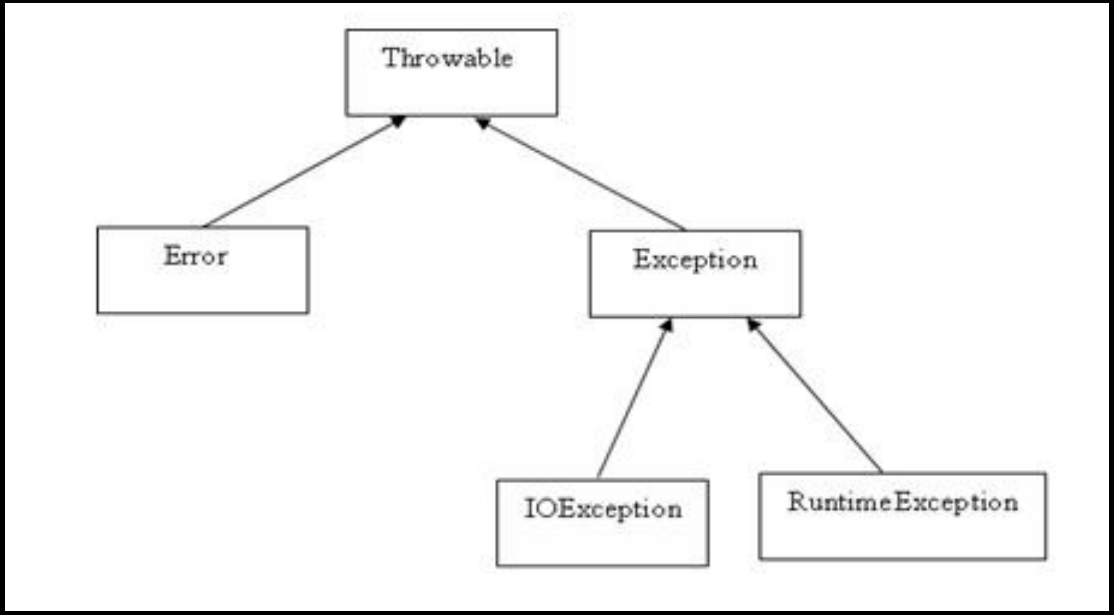
JAVA EXCEPTION HANDLING

An exception is a problem that arises during the execution of a program.

- **Checked exceptions**
- **Runtime exceptions**
- **Errors**
- Try, catch and finally are used to catch an exception

```
try {
        //Protected code
 }catch(ExceptionName e1) {
        //Handle Exception
}finally{
        //clean up code
}
```

JAVA EXCEPTION HANDLING



- Creating executable jars and distributing jar