**Introduction to java**

What does java do?

* Source
* Compile
* Output
* Virtual machines

Method 2:

Statement

Statement

statement

Method 1:

statement

Class file

**Source file**

Method 2:

Statement

Statement

statement

Method 1:

statement

Class file

**Source file**

Code structure of Java:

Method 2:

Statement

Statement

statement

Method 1:

statement

Class file

**Source file**

Anatomy of Java :

public class MyFirstApp {

public static void main(String args[]) {

System.out.println(“ I rule! “ );

}

}

Things that can be done in main:

* Statements: declaring,assigning,calling etc

Declaring and assigning a variable:

Int(decdlaring type of variable) number(variable) = 4(assigning a value to variable);

Int age = 26;

String name = “Anitha”;

String file = “JavaClass”;

* Loops : while and for

While loop ex:

Int x=5;

While (x> 2) {

System.out.println(“value of x is” + x);

X=x+1;

}

* Branching : if/else tests

If and else testing:

Int x=3;

If (x==4){

System.out.println(“x is 4”);

} else {

system.out.println(“x is not 3”);

}

**A trip to objectville**

The Importance of object oriented programming over procedural programming is explained by the use of classes.

Class : blueprint that defines how objects should behave

Super class: coding a class and declaring its behaviors so as not to duplicate the code for classes of same behaviors in future.

Sub class: has the same behavior as super class and are grouped below it.

Inheritance : linking sub classes to super classes.

While designing class, objects need

* State : input/instance variables

Things that the object knows/has.

* Behavior: output/methods

Things that the object does/executes.

Dot operator: accesses to object’s state and behavior

d.bark(); - calling dog d to bark

The two uses of main :

* To test real class
* To launch the Java application

**Exercise:**

Code Magnets:

class DrumKit {

boolean topHat = true;

boolean snare = true;

void playTopHat() {

system.out.println(“ding ding da-ding”);

}

}

class DrumkitTestDrive {

public static void main (String args[]) {

Drumkit d = new DrumKit;

d.playsnare();

d.snare = false;

d.playTopHat();

if (d.snare == true) {

d.playSnare();

}

}

}

**Know Your variables**

Variables:

There are two rules to declare a variable to the complier.

1. Variables must have a type
2. Variables must have a name

Eg: int age = 26

Int=data type

Age= variable name

26=assigned value

Primitive variable**:**

**These are** fundamental values.

Types:

* Int

1. Byte-8 bits
2. Short-16bits
3. Int-32bits
4. Long-64bits

* Char-16 bits
* Boolean

1. True
2. false

* Float

1. Float-32bits
2. Double-64bits

Java takes decimal values as double data type as default. Incase it must be changed to float, a “f” must be added when declaring and assigning a variable.

Eg: float f= 32.5 f;

Reference variable:

Reference variable/object reference variable are spaces allocated for ways to get to the object.

Eg: Dog myDog **= new** Dog();

Dog myDog = declaring a reference variable

New Dog(); = creating a object

**=** = linking object and the reference

**array:**

declaring a array

int[] nums;

nums=new int[4];

**nums[0]=6**

nums[1]=56

nums[2]=89

nums[3]=25

an array is an object even if it holds primitive data.

**How objects behave**

Arguments:

They can be passed into methods.  
d.bark(3) - 3 is the argument

We can get things back from a method:

Int givesecret () {

Return 42;

}

for the method to return values it should not be declared void.

Encapsulation:

It’s the method to protect data which are exposed to the public.so to hide the data we use the public and private access modifiers.

Rules for encapsulation:

* Mark instance variables private.
* Mark getters and setters public.

Declaring a instance variable:

Class poordog {

Private int size;

Private string name;

Public int getsize() {

Return sixze;

}

public string getname() {

return name;

}

difference between local and instance variables:

instance variables:

* They are declared inside a class; but not within a method.
* Instance variables always get a default value. Its usually null,0,00.00,false for various data types.

Local variables:

* They are declared inside a method.
* They must be initialized before use.
* They do not get a default value.

Comparing variables:

1. Use == to compare two primitives,or to see if two references refer to same object.
2. Use the equals () method to see if two different objects are equal.