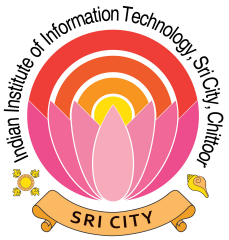


Monsoon 2019

Classes and Objects

Object Oriented Programming

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

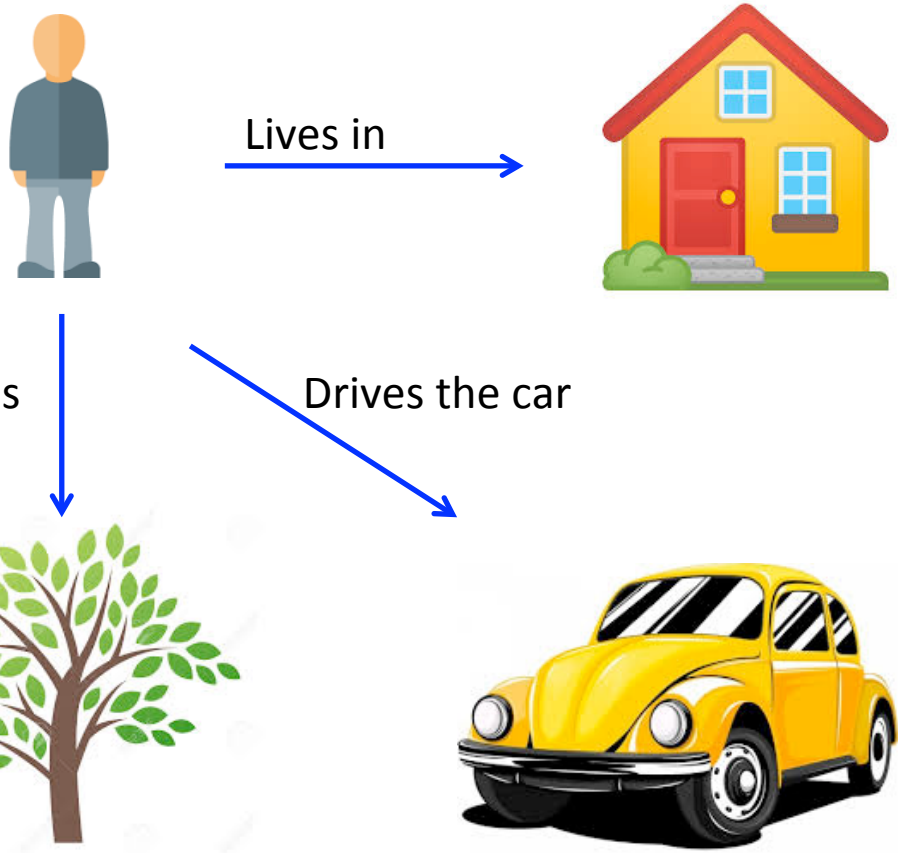
Sri City – 517 646, Andhra Pradesh, India

Recap: Model – An Example

- Various Objects and their relations

Recap: OO Model

- **Objects:**
  - Person
  - House
  - Car
  - Tree



- **Interaction**
  - Ram lives in the hou
  - Ram drives the car
  - Ram likes trees
  - and so on . . .

Objects and Classes

- **Object**
  - Object means an real world entity which has physical presence
  - It has two characteristics such as state and behavior
  - This can also be defined as instance of a class
  - Object takes space in memory

- **Class**
  - Class is nothing but a classification, also can be collection of objects
  - It is a logical entity
  - A class can also be defined as a blueprint which describes state/behaviors and objects can be created from it.
  - Class does not consume any space

Abstraction

- Hiding internal details and showing functionality is known as abstraction.
- For example phone call, we don't know the internal processing.
- In Java, we use the following to achieve abstraction:
  - abstract class and
  - interface

Inheritance

- When one object acquires all the properties and behaviors of another object (parent), this concept is known as Inheritance
- It creates a parent-child relationship b/w two classes
- It provides code reusability
- Provides a mechanism for organizing and structure of any software
- This is used to achieve runtime polymorphism

Polymorphism

- If one task is performed by different ways, it is known as polymorphism
- It refers the ability of a variable, object or function to take multiple forms
- For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- In Java, we use method overloading and method overriding to achieve polymorphism.

Encapsulations

- Binding (or wrapping) code and data together into a single unit are known as encapsulation
- For example capsule, it is wrapped with different medicines.
- A java class is the example of encapsulation.
- Java bean is the fully encapsulated class because all the data members are private here.

OOP Benefits

The benefits of using the object model are:

- ✧ It helps in faster development of software
- ✧ It is easy to maintain
- ✧ Suppose a module develops an error, then a programmer can fix that particular module, while the other parts of the software are still up and running
- ✧ It supports relatively hassle-free upgrades
- ✧ It enables reuse of objects, designs, and functions
- ✧ It reduces development risks, particularly in integration of complex systems

JAVA Programming

- A true OO language and the underlying structure of all Java programs is classes.
- Things represented in Java must be encapsulated in a class that defines an Object with "**state**" and "**behaviour**"
- Classes create objects and objects use methods to communicate between them
- They provide a convenient method for packaging a group of logically related data items and provides functionality
- A class essentially serves as a template for an object and behaves like a basic data type "**int**".
- **Important to understand:** how the Attributes and methods are defined and used in a class

OO Features

- Java is an Object-Oriented Language having the following concepts:
  - **Classes**
  - **Objects**
  - **Instance**
  - **Method**
  - **Message Parsing**
  - **Polymorphism**
  - **Inheritance**
  - **Encapsulation**
  - **Abstraction**

Objects in JAVA ?

- An entity that has **state** and **behaviour** is known as an object
  - **Examples:** Chair, bike, marker, pen, table, car etc
  - It can be physical or logical
- An object has three characteristics:
  - **State:** represents data (value) of an object
  - **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw and so on
  - **Identity (Internally used):**
    - Signature (unique) of the object
    - Object identity is typically implemented via a unique ID
    - The value of the ID is not visible to the external user
    - But, Internally by JVM to identify each object uniquely

States and Behaviour of Objects

- Consider an Object – **Pen**
- What are the states and Behaviours:
  - **States:**
    - Name: **Reynolds / Cello / Parker**
    - Color: **white**
    - Made in: **India / Europe**
    - Packed on: **19 July 2019**
  - **Behaviour:**
    - It is used to write, so **writing** is its behavior

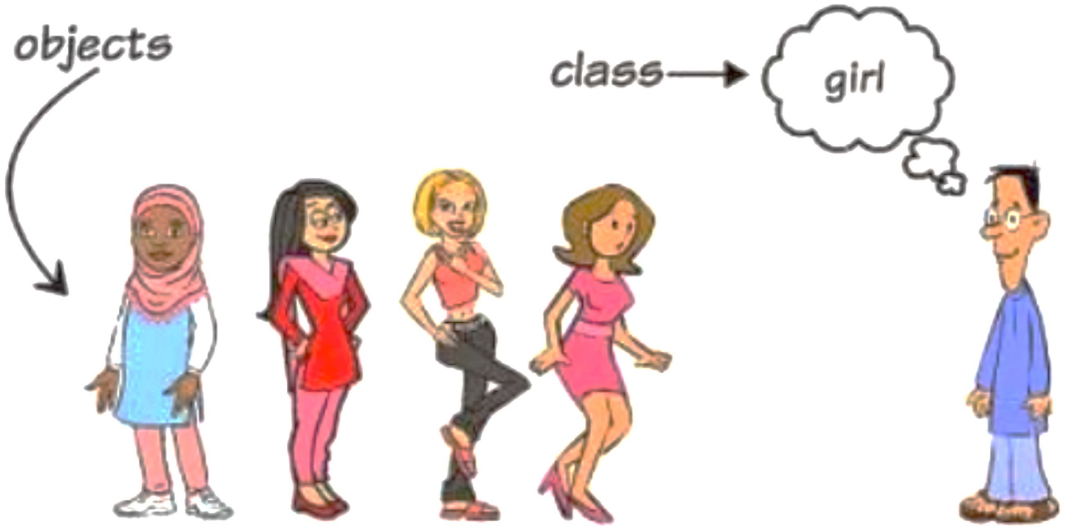
- Object is an Instance of a Class.
- Class is a template or blueprint from which objects are created

States / Behaviours - Examples

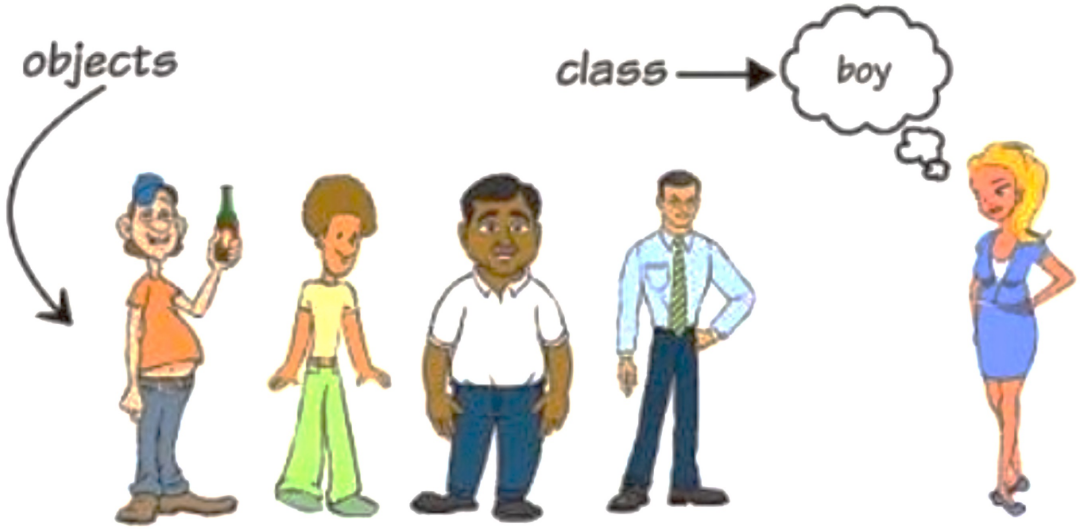
- **House**
  - **States:**
    - **Location, Exterior Color, House Number, Area**
  - **Behaviour:**
    - Close / Open the Door
- **Car**
  - **States:**
    - **Color, Make, Year**
  - **Behaviour:**
    - Climb Uphill, Accelerate, Slow Down and so on

Objects – A Fun Example

- Girls



- Boys



Class in JAVA

- A class is a group of objects that has common properties
  - It is a template from which objects are created

- A Class in JAVA can contain:
  - Data Members
    - Attribute - Values
  - Methods
  - Constructors
  - Blocks
  - Classes
  - Interfaces

Defining a Class

- In Java everything is encapsulated under classes. Class is the **core of Java language**
- A class defines new data type and can be used to create object of that type
- Object is an instance of class. You may also call it as **physical existence of a logical template class**
- A class is declared using **class** keyword.
- A class contain both **data** and **code** that operate on that data
- The data or variables defined within a class are called **instance variables** and the code that operates on this data is known as methods.

Class Diagram

- A class is a collection of
  - **Fields** (data) and
  - **Methods** (procedure or functions) that operate on that data

UML Class
Class Name
Attributes
Operation / Method

JAVA Class
Circle
center radius
area()

JAVA Naming Conventions

- Just a Guideline (Best Practice)!!

Name	Convention
Class	Should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc
Interface	Should start with uppercase letter and be an adjective e.g. Runnable, Remote
Variables	Should start with lowercase letter e.g. firstName, orderNumber etc
Methods	Should start with lowercase letter and be a verb, for example, actionPerformed(), main(), print(), println() and so on
Constants	Should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc
Package	Should be in lowercase letter e.g. java, lang, sql, util etc

JAVA Naming (contd.)

- CamelCase in java naming conventions
- Java follows camelcase syntax for naming the class, interface, method and variable
- If name is combined with two words, second word will start with uppercase letter always

- Examples:
  - actionPerformed()
  - firstName
  - ActionEvent
  - ActionListener

Class Declaration

- Declaration of class must start with the keyword class followed by the class name and class members are declared within braces.
- Syntax of Declaring Class

```
class class_name {
        // some data / fields

        // Some Methods
}
```

Note: A class is a **user defined data type**

Syntax of Declaring Class

```
access specifier class classname {
        type instance-variable1;
        …
        type instance-variableN;

        type methodname1(parameter-list) {
                // body of method 1
        }
        …
        type methodnameN(parameter-list) {
                // body of method N
        }
}
```

Class Explanation of Syntax

- Class Name

```
access specifier class classname {

}
```

- **Access Specifier:** Access modifiers (or specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members.
- **Access Specifier:** optional (**public**, **private**, **protected**)
- **class** is Keyword in Java used to create class in java.
- classname is Name of the User defined Class

Class Explanation – An Example

- Class Name


```
public class Square {

}
```

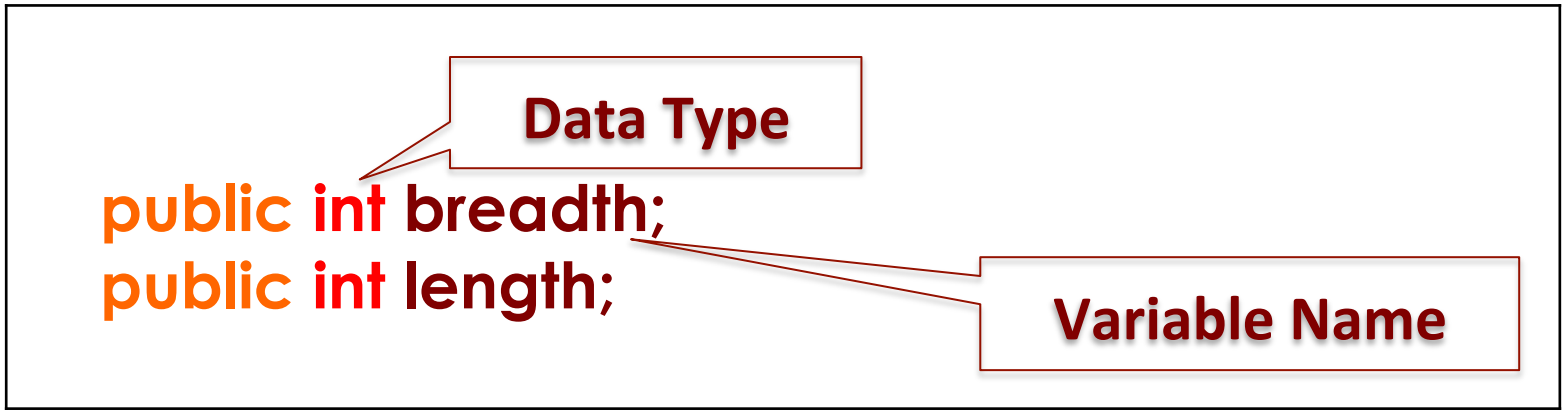
- **Access Specifier:** Access modifiers (or specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members.
- **Access Specifier:** optional (**public**, **private**, **protected**)
- **class** is Keyword in Java used to create class in java.
- classname is Name of the User defined Class

Class Instance Variables

```
type  instance-variable1;
type  instance-variable2;
 .
 .
 .
type  instance-variableN;
```

- Instance Variables are Class Variables of the class.
- When a number of objects are created for the same class, **same copy of instance variable is provided to all**
- **Instance variables have different value** for different objects.
- **Access Specifiers** can be applied to instance variable for example, public, private
- Instance Variable are also called as "**Fields**"

Variables – An example



- Instance Variables are Class Variables of the class.
- When a number of objects are created for the same class, **same copy of instance variable is provided to all**
- **Instance variables have different value** for different objects.
- **Access Specifiers** can be applied to instance variable for example, public, private
- Instance Variable are also called as "**Fields**"

Methods

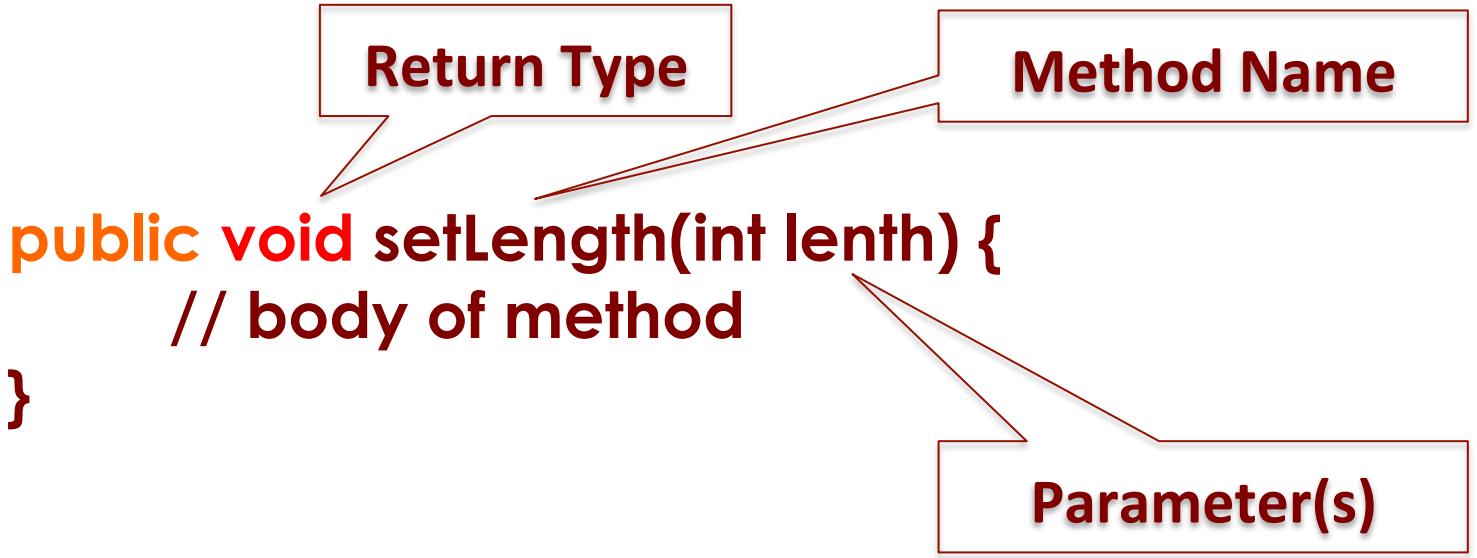
```
type  methodname1(parameter-list) {

        // body of method

}
```

- Instance Variables are Class Variables of the class.
- When a number of objects are created for the same class, **same copy of instance variable is provided to all**
- **Instance variables have different value** for different objects.
- **Access Specifiers** can be applied to instance variable for example, public, private
- Instance Variable are also called as "**Fields**"

Methods



- Above syntax is of **Class Methods**
- These methods are equivalent to function in C Programming Language
- Class methods **can be declared public or private**
- These methods are meant for operating on class data for example, **Class Instance Variables**
- Methods have **return type as well as parameter list**

A Complete Example

```
public class Square {
        // two fields
        public int breadth;
        public int length;

        // two methods
        public void setLength(int newValue) {
                length = newValue;
        }

        public void setBreadth(int newValue) {
                breadth = newValue;
        }
}
```

Rules for Java Class

- A class can have only public or default(no modifier) access specifier.
- It can be either abstract, final or concrete (normal class).
- It must have the class keyword, and class must be followed by a legal identifier.
- It may optionally extend one parent class. By default, it will extend java.lang.Object.
- It may optionally implement any number of comma-separated interfaces.
- The class's variables and methods are declared within a set of curly braces {}.
- Each .java source file may contain only one public class. A source file may contain any number of default visible classes.
- Finally, the source file name must match the public class name and it must have a .java suffix.

Advantages

- Visualize in terms of objects
  - OO models map to reality
  - OO models are: easy to develop and understand
- Objects are modeled on real world entities
  - This enables modeling complex systems of real world into manageable software solutions
- Advantage of OOPS
  - It provides data hiding
  - It provides data encapsulation
  - It provides data abstraction
  - It provides reusability of code
  - It provides easy code maintenance

Java Naming Conventions

- Java naming convention is a rule to follow as what to name your identifiers such as class, package, variable, constant, method, etc.
- But, **it is not forced to follow**. So, it is known as convention not rule

- The following are the key rules that must be followed by every identifier:
- The **name must not contain any white spaces**.
- The name should not start with special characters like & (ampersand), $ (dollar), _ (underscore).
- Advantages
  - By using standard Java naming conventions, you make your code easier to read for yourself and other programmers.
  - Readability of Java program is very important.
  - It indicates that less time is spent to figure out what the code does.

JAVA Prog – Remember 3 types

Example: Create a Banking Software with functions like

- Deposit
- Withdraw
- Show Balance

Class and interface

```
public class Employee  {
        //code snippet
}

interface Printable  {
        //code snippet
}
```

Package and constant

```
package com.bank;
class Employee  {
        //code snippet
}
class Employee  {
        //constant
        int MIN_AGE = 18;
        //code snippet
}
```

Variables and Methods

```
class Employee {
        //variable
        int id;
        //code snippet
}

class Employee  {
        //method
        void draw()  {
        //code snippet
        }
}
```

Java keywords

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while	true	false

Java Operator Precedence

Precedence	Operator	Type	Associativity
1	()	Parentheses	Left to Right
	[]	Array subscript	
	.	Member selection	
2	++	Unary post-increment	Right to left
	--	Unary post-decrement	
3	++	Unary pre-increment	Right to left
	--	Unary pre-decrement	
	+	Unary plus	
	-	Unary minus	
	!	Unary logical negation	
	~	Unary bitwise complement	
	(type)	Unary type cast	
4	*	Multiplication	Left to right
	/	Division	
	%	Modulus	
5	+	Addition	Left to right
	-	Subtraction	
6	<<	Bitwise left shift	Left to right
	>>	Bitwise right shift with sign extension	
	>>>	Bitwise right shift with zero extension	

Java Operator Precedence

Precedence	Operator	Type	Associativity
7	<	Relational less than	Left to right
	<=	Relational less than or equal	
	>	Relational greater than	
	>=	Relational greater than or equal	
	instanceof	Type comparison (objects only)	
8	==	Relational is equal to	Left to right
	!=	Relational is not equal to	
9	&	Bitwise AND	Left to right
10	^	Bitwise exclusive OR	Left to right
11	|	Bitwise inclusive OR	Left to right
12	&&	Logical AND	Left to right
13	||	Logical OR	Left to right
14	? :	Ternary conditional	Right to left
15	=	Assignment	Right to left
	+=	Addition assignment	
	-=	Subtraction assignment	
	*=	Multiplication assignment	
	/=	Division assignment	
	%=	Modulus assignment	

Exercises

- **Create Geometric Objects**
  - Perform Basic Operations
  - Apply Transformation
  - Perform getter and setter
  - Extending the Object to Other Shapes

- **Bank Application**
  - Employee
  - Customer
  - ATM
  - Account details
    - Balance enquiry

Assignments / Penalties

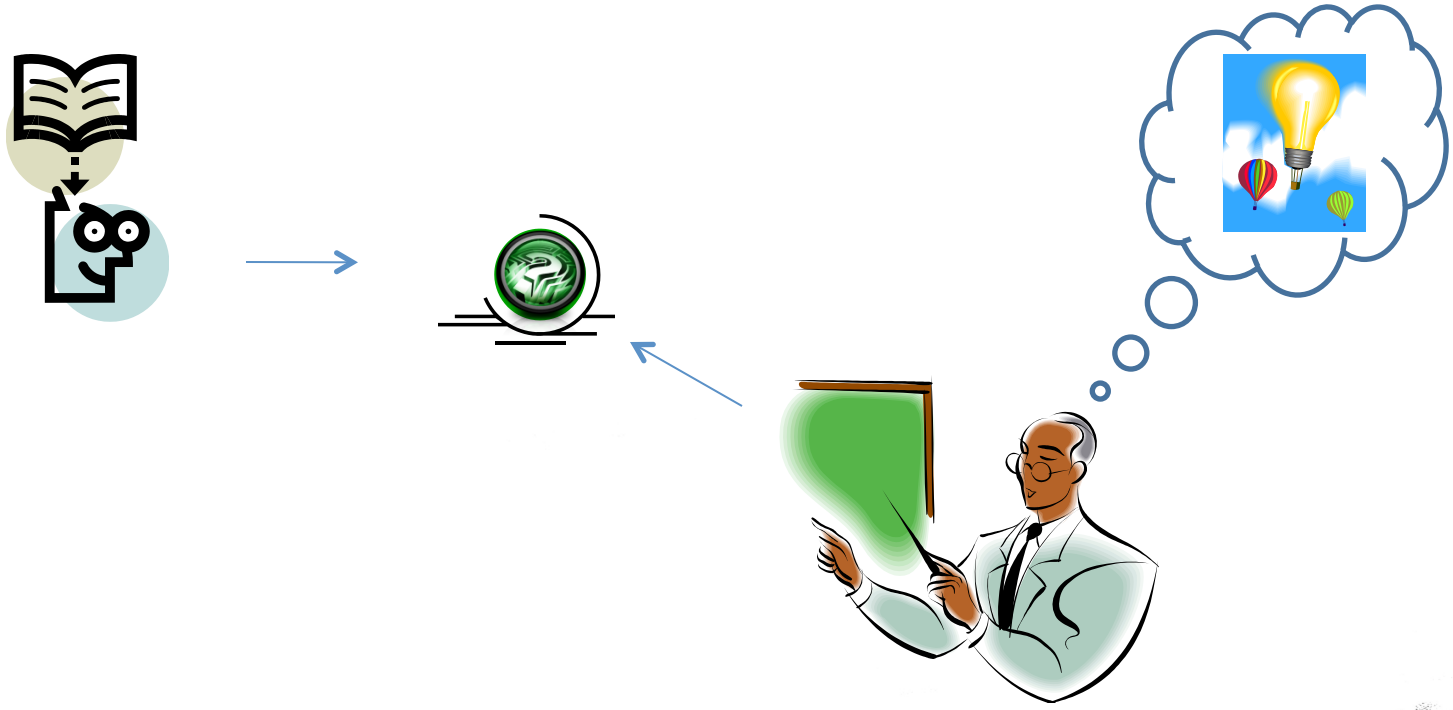
- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties

- Penalties would be heavy for those who involve in:
  - **Copy and Pasting** the code
  - **Plagiarism** (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is **unable to explain his own solution**, it would be considered as a "copied case" !!
  - **Any other unfair means** of completing the assignments

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
  (email is the best way to reach me faster)

Thanks …



… Questions ???