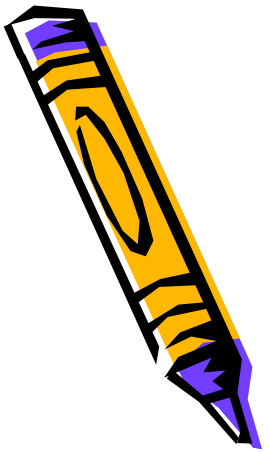
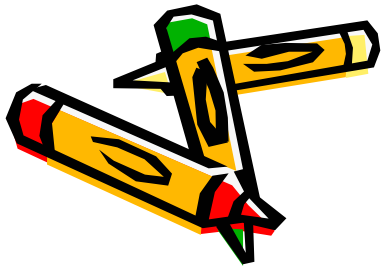
Introduction to Classes and Objects

Chapter 3

Objectives

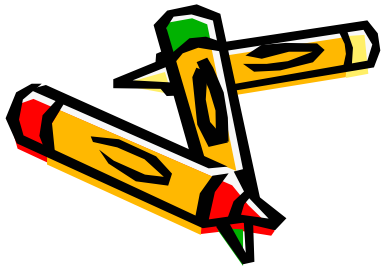


- Define a class with data members
- Differentiate the local and instance variables
- Distinguish private and public data members
- Define a class with methods
- Distinguish private and public methods
- Define and use value-returning methods
- Pass both primitive data and objects to a method



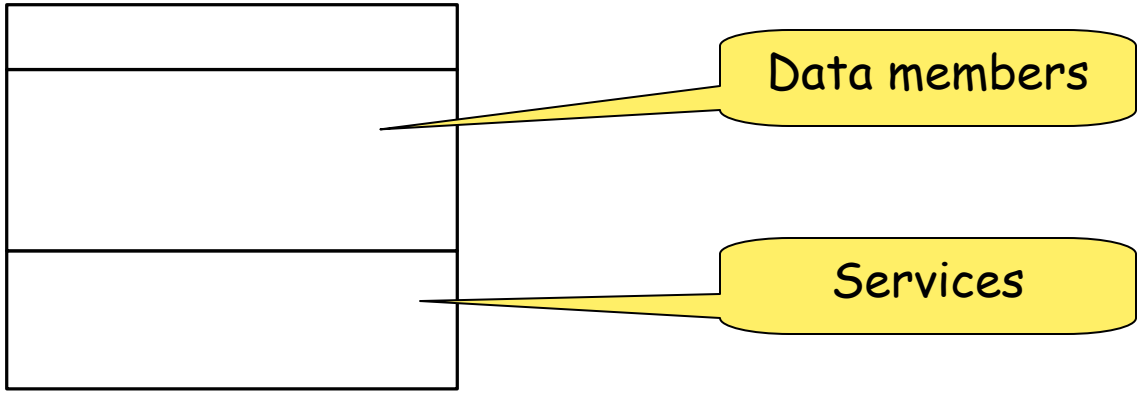
Programmer-Defined Classes

- Defining our own classes customized for our applications.
- Learning how to define our own classes is the first step toward mastering the skills necessary in building large programs.
- Classes we define ourselves are called programmer-defined classes.



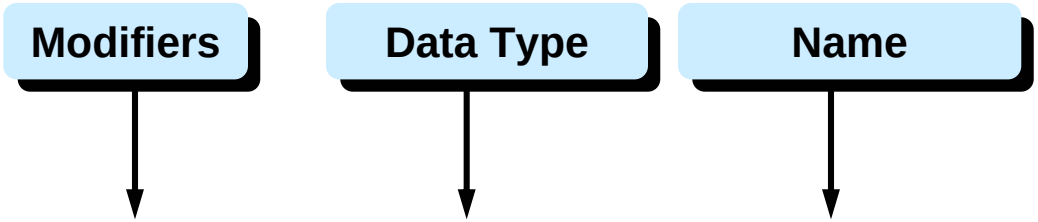
Class scope

- A class has :
  - A name
  - variables or data members
  - methods
- Members are accessible to all class methods

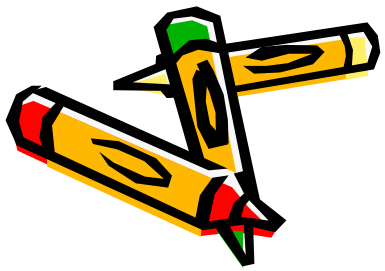


Data Member

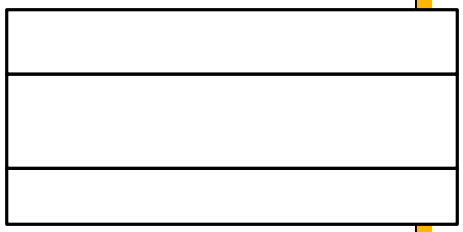
```
<modifiers>  <data type> <name> ;
```



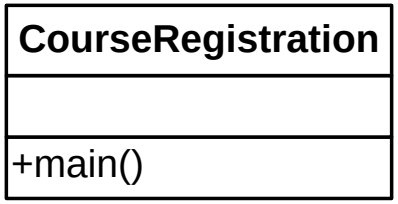

```
public      String     studentName ;
```

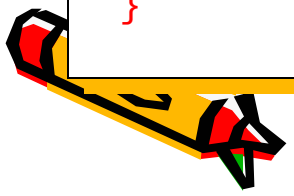


```
class Course {

    // Data Member
    public String studentName;
    public String courseCode ;

}
```



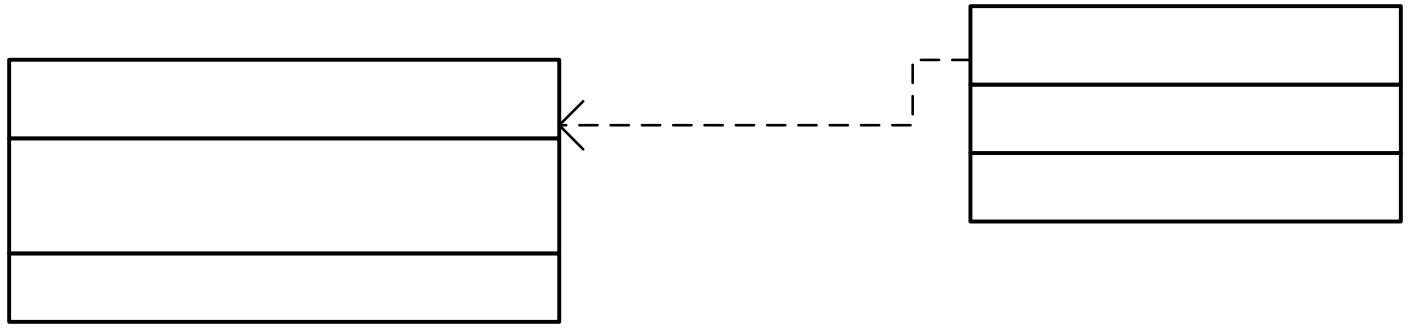
```
public class CourseRegistration {
    public static void main(String[] args) {
        Course course1, course2;
//Create and assign values to course1
        course1 = new Course( );
        course1.courseCode= new String("CSC112");
        course1.studentName= new String("Majed AlKebir");
//Create and assign values to course2
        course2 = new Course( );
        course2.courseCode= new String("CSC107");
        course2.studentName= new String("Fahd AlAmri");
        System.out.println(course1.studentName + " has the course "+
                                                course1.courseCode);
       System.out.println(course1.studentName + " has the course "+
                                                course1.courseCode);
    }
}
```





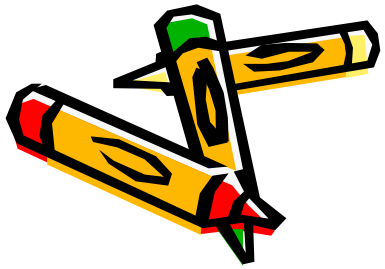
Practical hint

- Class `course` will not execute by itself
  - It Does not have method `main`
  - CourseRegistration, which has method `main`, creates 2 *instances of the class Course: course1 and course2* and uses `them as` objects
- CourseRegistration is considered as **client** for Course



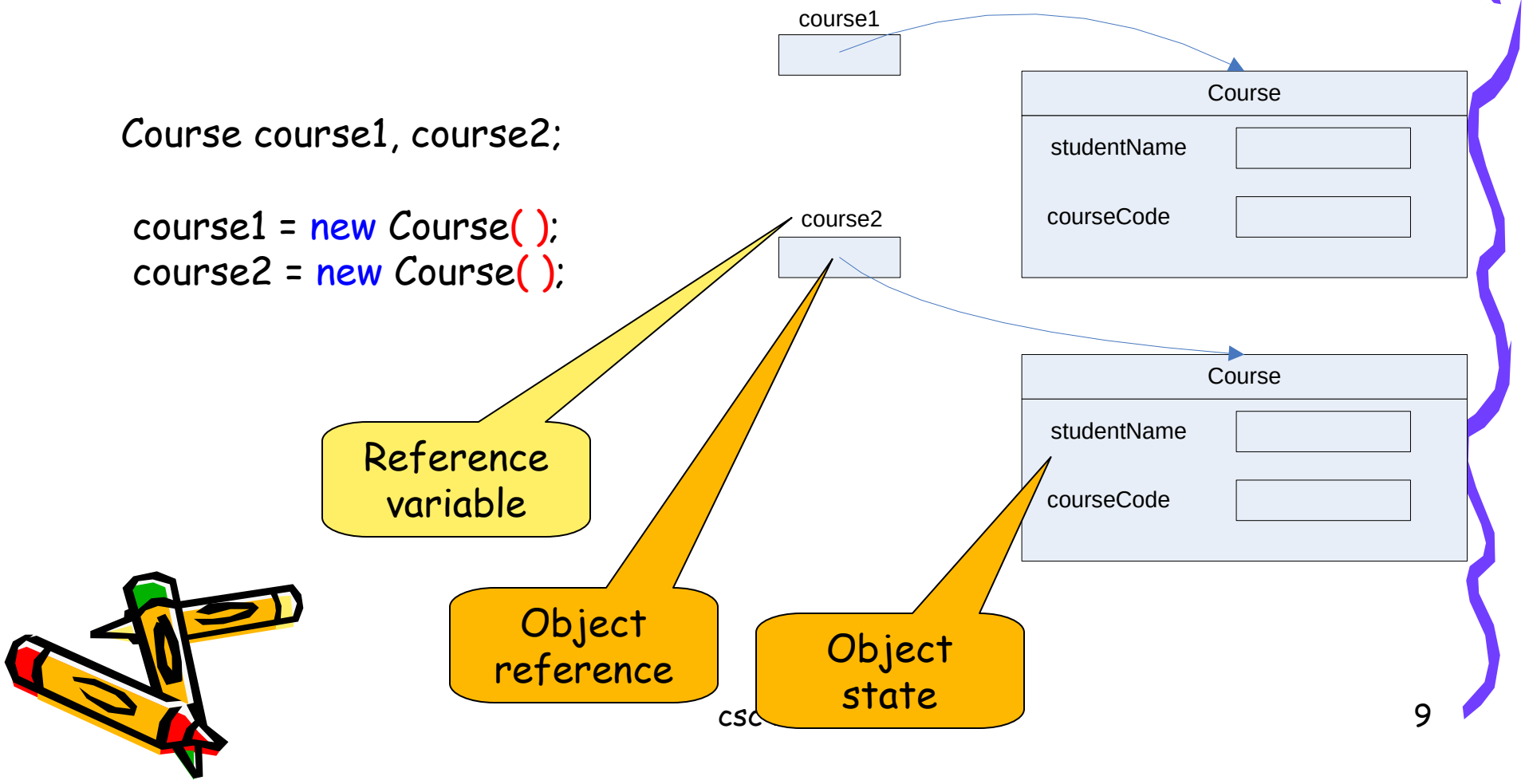
Primitive vs. Reference

- Numerical data are called *primitive data types*.
- Objects are called *reference data types*, because the contents are addresses that refer to memory locations where the objects are actually stored.



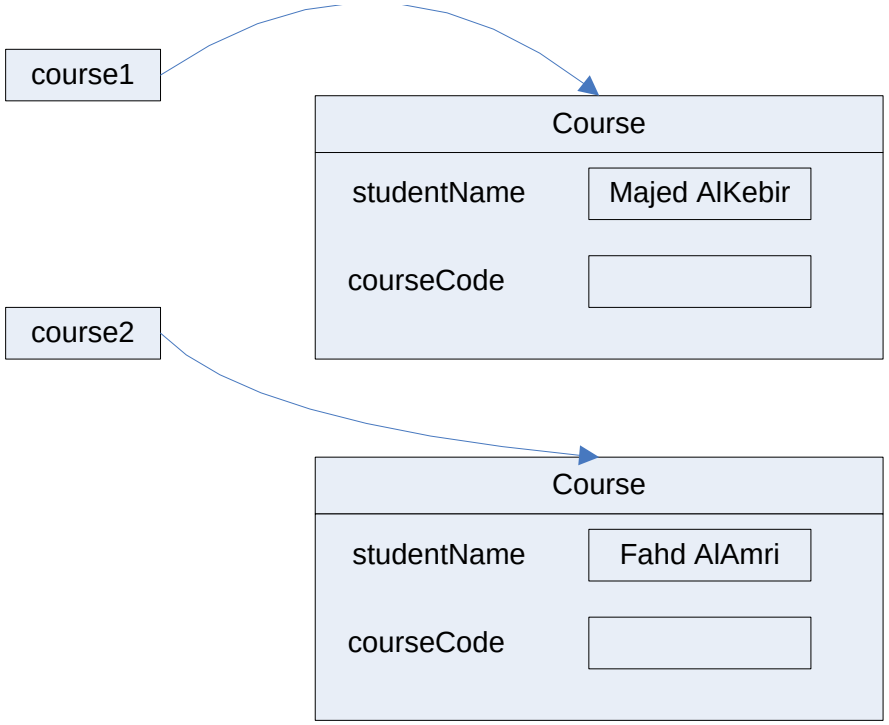
Object declarations: Instance variables

- Once the Student class is **defined**, we can create several instances.



Instances

- course1.StudentName=new String("Majed AlKebir");
- course2.StudentName= new String("Fahd AlAmri ");



Assigning Objects

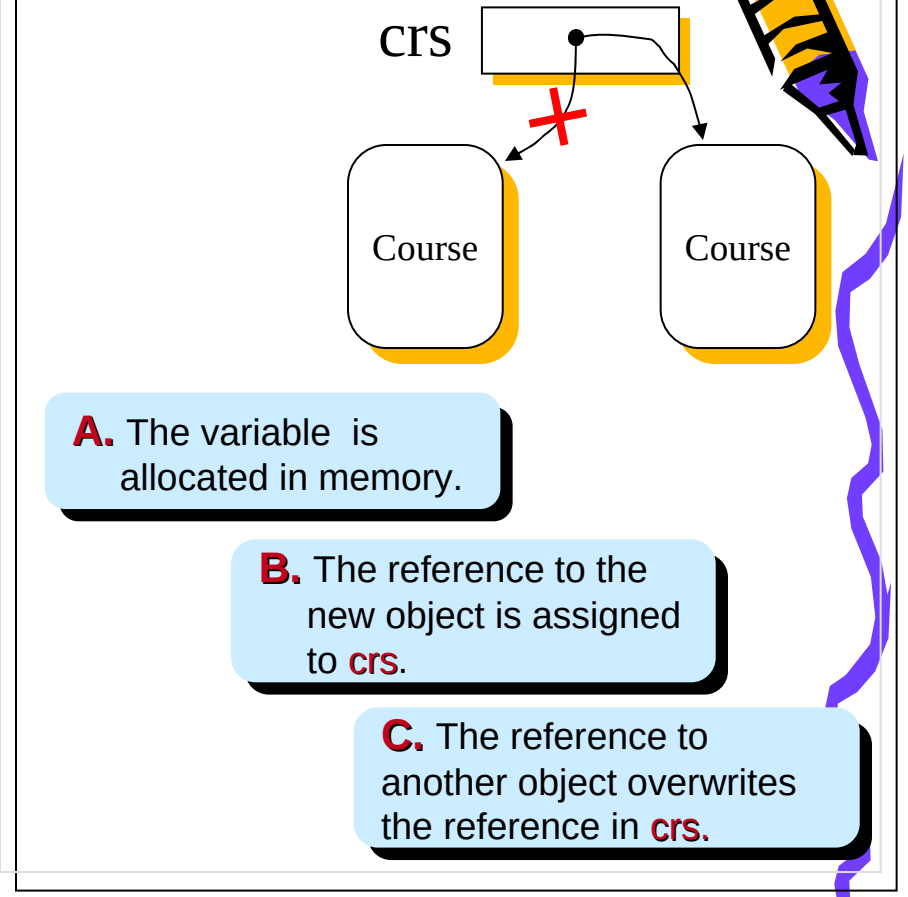
```
Course crs;

crs = new Course( );

crs = new Course( );
```

A

B

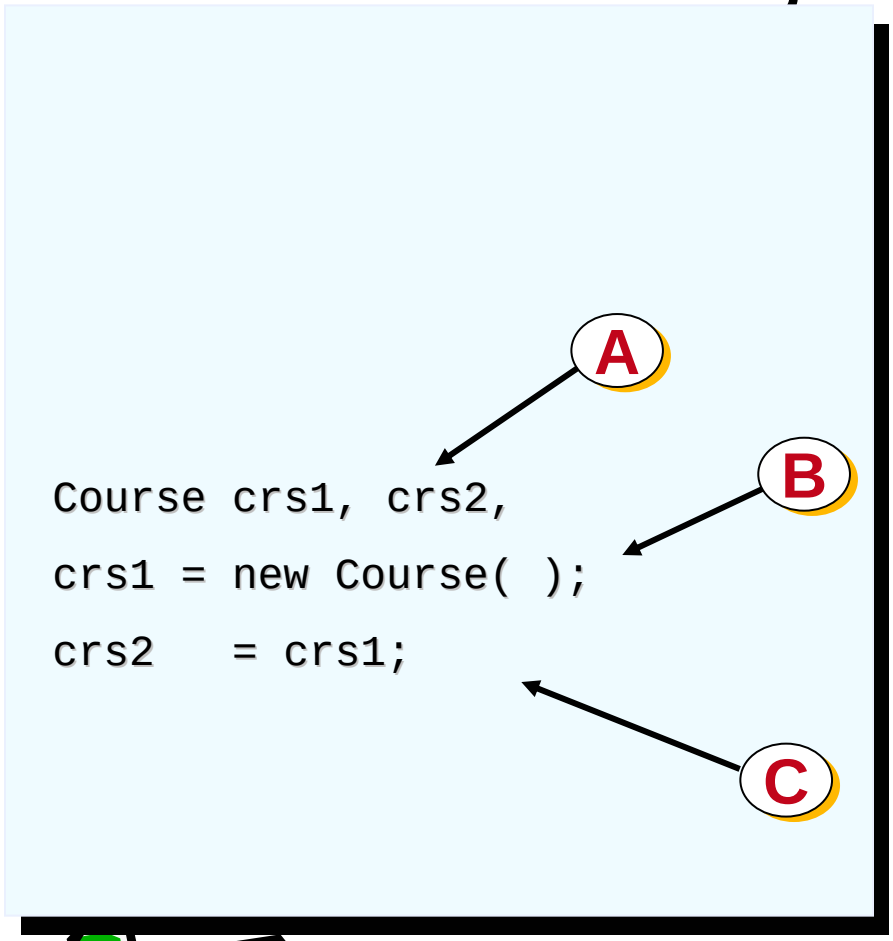
C

Code

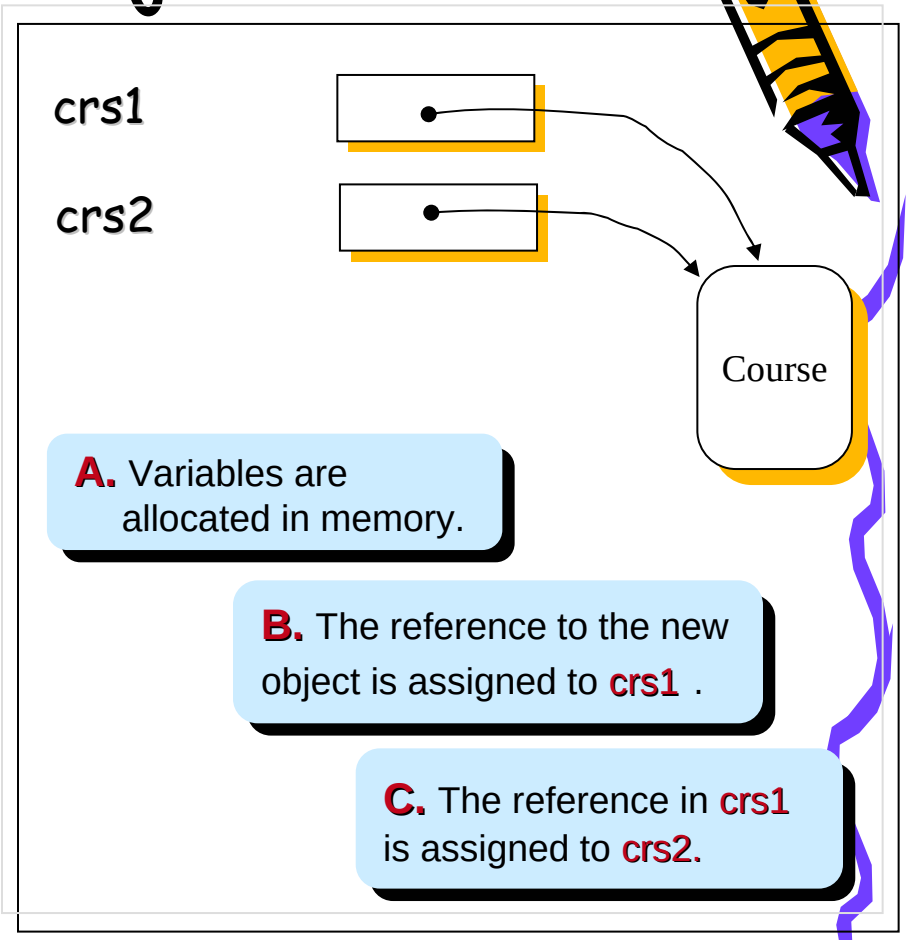


State of Memory

Having Two References to a Single Object

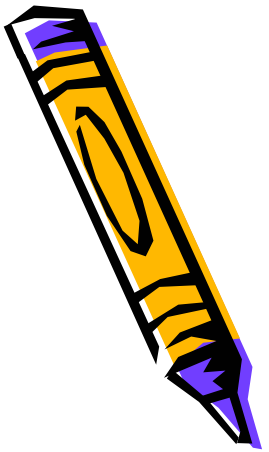


Code

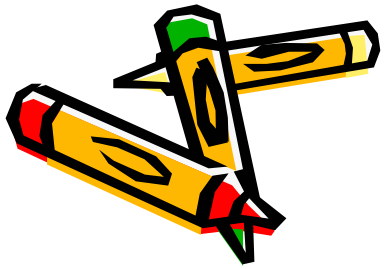


State of Memory

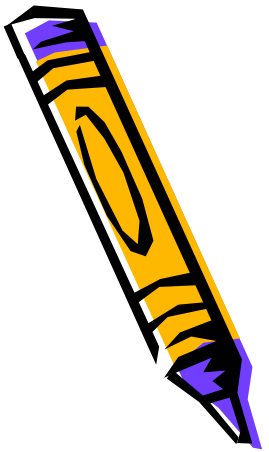
Instance variable



- Attribute variables are also called member or instance variables
- When an object is instantiated from a class, these variables contain data specific to a particular object instance of the class



The use of variable



- Variables are used:
  - To hold unique attribute data for an object instance.
  - To assign the value of one variable to another.
  - To represent values within a mathematical expression.
  - To print the values to the screen.
  - To hold references to other objects