SENG1110/SENG6110
Object Oriented Programming

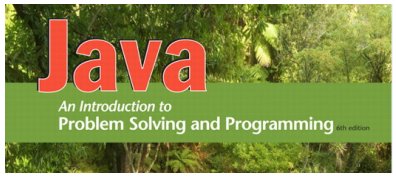
THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

www.newcastle.edu.au

Lecture 5

Classes and Methods – Student example



Example: Student class

- What are the instance variables (data) and methods of each object of this class?
- Instance variables: each student contains
  - a name
  - three test scores
- Methods: the student needs to respond to (it needs to have the methods):
  - setName
  - getName
  - setScore
  - getScore
  - getAverage

Example: Student class

- Variables: types?

```
String name
int test1, test2, test3;
```

- Methods: the signature (return, name and parameters)

```
void setName(String)
String getName()
void setScore(int whichTest, int testScore)
int getScore(int whichTest)
int getAverage()
```


Student class – instantiate an object - `new`

- Before write/define the class student, let's see how to instantiate an object from a class.
- When you need to use a variable
  - first you declare

```
int number;
```

  - Next you assign a value to it

```
number = 4;
```

- When you have class (Student in our example), you need to do the same, but you need to use the word `new`:

```
Student s1;
s1 = new Student();
```

  - Or

```
Student s1 = new Student();
```

Student class – instantiate an object - `new`

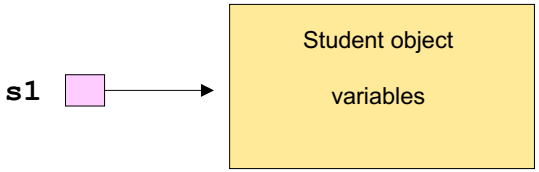
- Remember, when you do

```
Student s1;
s1 = new Student();
```

  - Or

```
Student S1 = new Student();
```

- The object s1, in fact is a reference to a student object



Student class – using the methods

- Just to remember…when you declare

```
String str = "Hey Joe!";
```

- `str` is an object from `String` class.
- `String` class has several methods (for example, `length` that we can use).
- We don't know how the `length` method is implemented, but we can use it if we know its signature (name, return and parameters). Its signature is

```
int length()
```

- Then I can use, for example, as:

```
System.out.print(str.length());
```


Student class – using the methods

- Let's see some examples of how use the methods from Student class:
- Suppose we have the object created as

```
Student s1,s2,s3;
s1 = new Student();
s2 = new Student();
```

- s1 has the reference of an object student
- s2 has the reference of another object student

Student class – using the methods

- `void setName(String)`

```
s1.setName("Joao");
s2.setName("Maria");
```

  - s1 has the reference of an object student which has the name "Joao"
  - s2 has the reference of an object student which has the name "Maria"

- `String getName()`

```
str = s1.getName();
```

  - Suppose str was declared as String
  - str will hold "Joao"

Student class – using the methods

- `setScore(int whichTest, int testScore)`

```
s2.setScore(2,89);
```

  - s2 has the reference of an object student which has the name = “Maria” and test2 = 89.

- `int getScore(int whichTest)`

```
x = s2.getScore(2);
```

  - s2 has the reference of an object student which has the attribute name = “Maria” and test2 = 89 (we did this before)
  - Suppose the variable x was declare as an int.
  - x will receive 89.

Student class – using the methods

- `int getAverage()`

```
s2.setScore(1,50);
s2.setScore(2,60);
s2.setScore(3,70);
x = s2.setAverage();
```

  - s2 has the reference of an object student which has the name = “Maria” (done before), test1 = 50, test2 = 60, test3 = 70.
  - Suppose the variable x was declare as an int.
  - x will receive 60

Student class – using the methods


```
Student s1,s2,s3;
s1 = new Student();
s2 = new Student();

s3 = s2   //s2 and s3 are references to the same student
s1.setName(“Joao”);
s2.setName(“Maria”);
str = s3.getName ();// str will receive “Maria”
```


Student class – implementation


```
public class Student
{
   // Instance variables: name, test1, test2, test3
   …
   // Constructor method
   …
   // other methods
    void setName(String)
    String getName()
    setScore(int whichTest, int testScore)
    int getScore(int whichTest)
    int getAverage()
}
```

Student class – implementation


```
public class Student
{   /**
     Instance variables
     Each student object will have a name and three test scores
    */
    private String name;              //Student name
    private int test1;                //Score on test 1
    private int test2;                //Score on test 2
    private int test3;                //Score on test 3

    /**
     Set a student's name
    Preconditions  -- nm is not empty
    Postconditions -- name has been set to name
    */
    public void setName (String nm)
    {
          name = nm;
    }
    /** Get a student's name
    Preconditions  -- none
    Postconditions -- returns the name
    */
    public String getName ()
    {
          return name;
    }
```


Student class – implementation


```
    /** Set the score on the indicated test
     Preconditions  -- 1 <= i <= 3
                       -- 0 <= score <= 100
     Postconditions -- test i has been set to score
    */

    public void setScore (int i, int score)
    {
                    if      (i == 1) test1 = score;
          else if (i == 2) test2 = score;
          else             test3 = score;
    }

    /** Get the score on the indicated test
        Preconditions  -- none
        Postconditions -- returns the score on test I
    */
    public int getScore (int i)
    {
    if      (i == 1) return test1;
          else if (i == 2) return test2;
          else             return test3;
    }
```


Student class – implementation


```
    /** Compute and return a student's average
      Preconditions  -- none
      Postconditions -- returns the average of the test scores
    */
    public int getAverage()
    {
    int average;
          average = (int) Math.round((test1 + test2 + test3) / 3.0);
          return average;
    }
}
```


Student class – implementation

- Now we have the class Student implemented.
- Next you have to compile your class using

`javac Student.java`

And…now…what?

- Notice that
  - It doesn't have main method.
  - We didn't input or output data…

Student class – implementation

- Now we have a new class that we can use in another application.
- Let's see one example

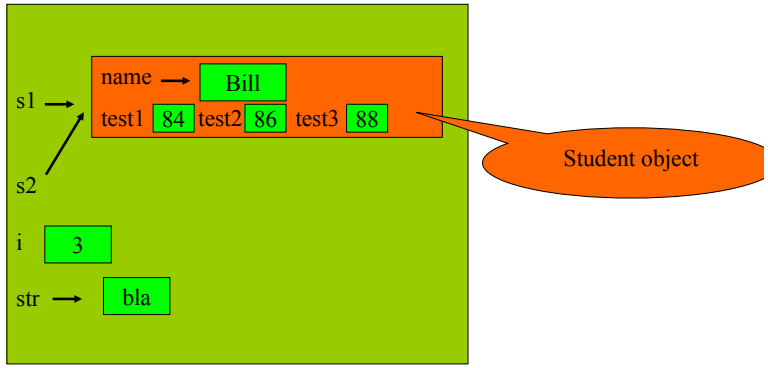
Student class – used in another code – Example1


```
import java.util.*;

public class TestStudent
{
     public static void main (String[] args)
    {
          Student s1, s2;
          String str="bla";
          int i=3;

          s1 = new Student();      // Instantiate a student object
          s1.setName ("Bill");     // Set the student's name to "Bill"
          s1.setScore (1,84);      // Set the score on test 1 to 84
          s1.setScore (2,86);      //               on test 2 to 86
          s1.setScore (3,88);      //               on test 3 to 88
          System.out.println("\nHere is student s1\n");
          System.out.println("\nname:"+s1.getName());
          System.out.println("\nTest 1 = "+s1.getScore(1));
          System.out.println("\nTest 2 = "+s1.getScore(2));
          System.out.println("\nTest 3 = "+s1.getScore(3));
          System.out.println("\nAverage = "+s1.getAverage());
          s2 = s1;               // s1 and s2 now refer to the same object
          s2.setName ("Ann");    // Set the name through s2
          System.out.println ("\nName of s1 is now: " + s1.getName());
    }
}
```


Student example1



Student class – used in another code – Example2


```
import java.util.*;

public class TestStudent
{
    public static void main (String[] args)
    {
          Scanner console = new Scanner(System.in);

          private Student student1;
          private Student student2;
          String s;
          int t;

          student1 = new Student();
          student2 = new Student();

          s = console.next();
          student1.setName (s);
          t = console.nextInt();
          student1.setScore (1, t);
          student1.setScore (2, console.nextInt());
          student1.setScore (3, console.nextInt());

          student2.setName (keyboard.readLine());
          student2.setScore (1, console.nextInt());
          student2.setScore (2, console.nextInt());
          student2.setScore (3, console.nextInt());
    }
}
```

student1 and **student2** are variables of type Student (objects, reference type)

student1 and **student2** are instantiated. The constructor from Student class is called

We are accessing **student1** using the methods

We are accessing **student2** using the methods

Student example 2