Lecture 18: Object Oriented Programming cont.
Recap to the previous lecture!

• What is an object?
• What is a class?
• How to create class and objects?
• Public vs Private.
• Linking OOP with arrays and loops.
Lecture 18 : OOP

• Constructors
public class student {

    private String name;
    private float gpa;

    void set_name (String new_name){
        name = new_name;
    }

    String get_name (){  
        return name;
    }

    void Set_gpa (float new_gpa){
        gpa = new_gpa;
    }

    float get_gpa (){  
        return gpa;
    }

    void print_Info (){  
        System.out.println("Student name "+name+" with gpa "+gpa);
    }

}
public class mainclass {

    public static void main(String[] args) {

        //define objects
        student[] student_array = new student[3];
        for(int i=0; i<student_array.length; i++){
            student_array[i] = new student();
        }

        //setting values
        student_array[0].set_name("Alice");
        student_array[0].set_gpa(90);
        student_array[1].set_name("Adam");
        student_array[1].set_gpa(80);
        student_array[2].set_name("Mickle");
        student_array[2].set_gpa(89);

        //printing values
        for(int i=0; i<student_array.length; i++){
            student_array[i].print_Info();
        }
    }
}
public class mainclass {

    public static void main(String[] args) {

        student[] student_array = new student[2];
        for(int i=0;i<student_array.length;i++){
            student_array[i] = new student();
        }

        Scanner input = new Scanner(System.in);

        for(int i=0;i<student_array.length;i++){
            System.out.println("Insert name for student "+i);
            String name = input.next();
            student_array[i].set_name(name);

            System.out.println("Insert gpa for student "+i);
            float gpa = input.nextFloat();
            student_array[i].set_gpa(gpa);

        }

        for(int i=0;i<student_array.length;i++){
            student_array[i].print_Info();
        }
    }
}
Class Student with one private variable gpa

```java
public class Student {
    private float gpa;

    void set_gpa (float new_gpa){
        gpa = new_gpa;
    }

    float get_gpa (){  
        return gpa;
    }
}
```
Getting value before setting it, returns the default value according to the data-type

```java
student std1 = new student();
System.out.println("GPA is "+std1.get_gpa());
```

Output: GPA is 0.0
Extending student class to include another private variable, name

```java
public class student {

    private String name;
    private float gpa;

    void set_name (String new_name){
        name = new_name;
    }

    String get_name (){:
        return name;
    }

    void set_gpa (float new_gpa){
        gpa = new_gpa;
    }

    float get_gpa (){:
        return gpa;
    }

}
Getting value before setting it, returns the **default** value according to the data-type

```java
student std1 = new student();
System.out.println("name is "+std1.get_name());
System.out.println("GPA is "+std1.get_age());
```

Output:

```
run:
name is null
GPA is 0
```
What is a java constructor

• A way programmer can initialize class members (variables) during creating an object.

• The constructor is called once for each object.
Student std1 = new student();

• The compiler is creating object named “std1”, of type student.

• The compiler is calling something called constructor using “student()”.
• Class constructor is a special type of functions, \textit{first form}:

\begin{verbatim}
Public Classname {

    Class variables (members)

    Classname() { //class constructor

    }

    Class functions (methods)

}
\end{verbatim}
public class student {

    private String name;

    void set_name (String new_name){
        name = new_name;
    }

    String get_name (){ 
        return name;
    }

}
• The compiler will check for a constructor in class.

• If not found, the compiler will create a “default constructor” and call it.

• Default constructor is a constructor with no parameters.
public class student {

    private String name;

    student() { // default constructor
    }

    void set_name (String new_name) {
        name = new_name;
    }

    String get_name () {
        return name;
    }

}
• This default constructor, is an empty constructor.

• Nothing is initialized inside the scope of this constructor.

• The compiler will call this empty constructor while creating object of this class.
The programmer can define his constructor as follows:

```java
public class student {

    private String name;

    student() { //default constructor
        name = "default name";
    }

    void set_name (String new_name) {
        name = new_name;
    }

    String get_name () {
        return name;
    }

}
```
The compiler will use student constructor while creating Object std1

```java
student std1 = new student();
System.out.println("name is "+std1.get_name());
```

Output:
```
run:
name is default_name
```
• Constructors are called once, during object creation.

```java
student std1 = new student();
```
public class Student {

    private String name;

    private int age;

    Student() { // default constructor
        name = "default name";
        age = 18;
    }

    void setName(String new_name) {
        name = new_name;
    }

    String getName() {
        return name;
    }

    void setAge(int new_age) {
        age = new_age;
    }

    int getAge() {
        return age;
    }
}
student std1 = new student();
System.out.println("name is "+std1.get_name());
System.out.println("GPA is "+std1.get_age());
• The output will be:

    run:
    name is default_name
    GPA is 18
student std1 = new student();
System.out.println("name is "+std1.getName());
System.out.println("GPA is "+std1.getGPA());
student std2 = new student();
System.out.println("name is "+std2.getName());
System.out.println("GPA is "+std2.getGPA());
• The output will be:

```python
run:
name is default_name
GPA is 18
name is default_name
GPA is 18
```
• The first constructor type (no parameters – default constructor), all objects are initialized the same way.

• The second for of constructor, is a constructor with parameters.

• Programmer can use this type to initialize the class variables with specific values for each object.
• Class constructor is a special type of functions, *second form*:

Public Classname { 

  Class variables (members) 

  Classname( some parameters){  //class constructor 
  
  } 

  Class functions (methods) 

  }
public class student {

    private String name;

    private int age;

    student(String start_name, int start_age){
        //constructor
        name = start_name;
        age = start_age;
    }

    void set_name (String new_name){
        name = new_name;
    }

    String get_name (){  
        return name;
    }

    void set_age (int new_age){
        age = new_age;
    }

    int get_age (){  
        return age;
    }
}
student std1 = new student("adam",18);
System.out.println("name is "+std1.get_name());
System.out.println("GPA is "+std1.get_age());
• The compiler –during object creation- is looking for constructor that takes 2 parameters (The first is string and second is integer).
• The output will be:

run:
name is adam
GPA is 18
student std1 = new student("adam", 18);
System.out.println("name is "+std1.get_name());
System.out.println("GPA is "+std1.get_age());

student std2 = new student("alice", 16);
System.out.println("name is "+std2.get_name());
System.out.println("GPA is "+std2.get_age());
• The output will be:

```python
run:
name is adam
GPA is 18
name is alice
GPA is 16
```
• Class constructor is a special type of functions, *third form*:

```
Public Classname {
    Class variables (members)
    Classname() { //first constructor
        }
    Classname(some parameters) { //second constructor
        }
    }
    Class functions (methods)
    }
```
public class student {
    private String name;
    private int age;

    student(){
        name = "default_name";
        age = 18;
    }

    student(String start_name, int start_age){ //constructor
        name = start_name;
        age = start_age;
    }

    void set_name (String new_name){
        name = new_name;
    }

    String get_name (){    
        return name;
    }

    void set_age (int new_age){
        age = new_age;
    }

    int get_age (){    
        return age;
    }
}
Creating two objects, with the different constructors of Student class

```java
student std1 = new student();
System.out.println("name is "+std1.get_name());
System.out.println("GPA is "+std1.get_age());

student std2 = new student("const_name",20);
System.out.println("name is "+std2.get_name());
System.out.println("GPA is "+std2.get_age());
```
Output of the two objects, created with the two different constructors.

run:
name is default_name
GPA is 18
name is const_name
GPA is 20
Try this!

• Create structure for 2D point, each point contains x and y.

• Create the different suitable constructors.

• Create the suitable set/get pairs.

• Create function that prints the point in the following format: ( x , y )
Try this!

- Create a structure for a boat in the sea.
- The boat has the following parameters: Name, direction (char), speed (int).
- Create the different constructors required.
- Create the different pairs of set/get.
- Create function that returns the value of these parameters in a readable format.