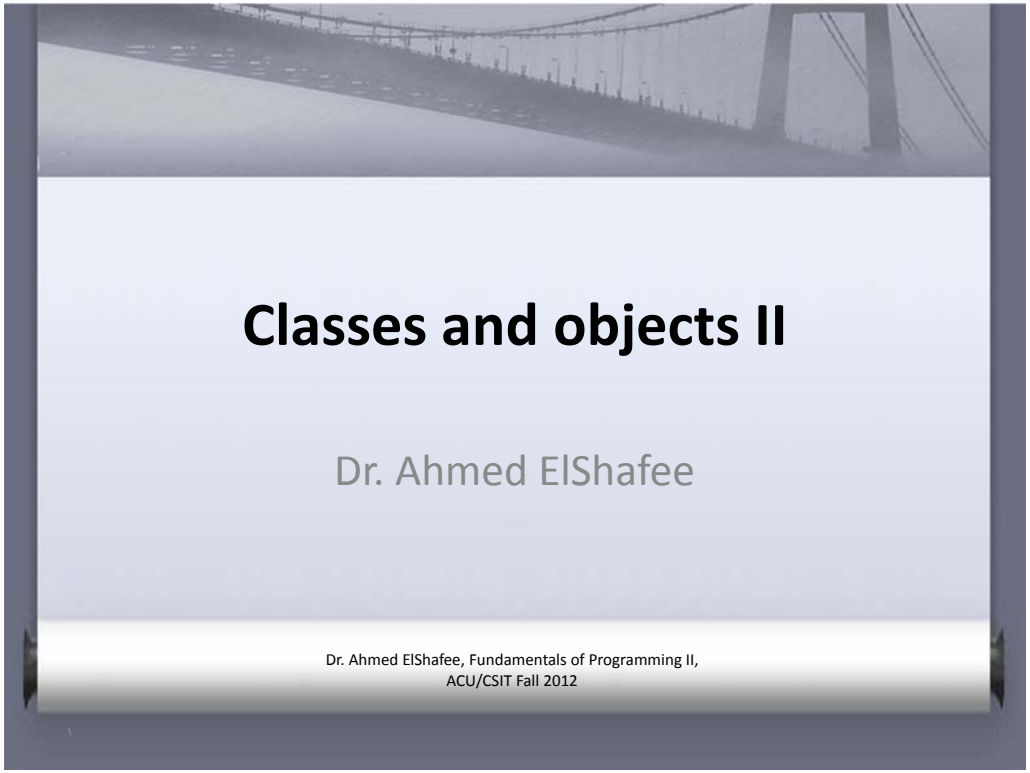

Classes and objects II

Dr. Ahmed ElShafee

Agenda

1. *
2. *
3. *

Method Overloading

- Java does not allow two variables to have the same name in the same method.
- Although two methods should have unique names in the same program, a class can have different methods with the same name if you follow some rules.
- The ability to have various methods with the same name in the same program is referred to as method overloading.
- To perform overloading, the methods must have different numbers or different type(s) of arguments.

lecture0514

Check lab manual

Class Construction

- A constructor is a special method that is created when the object comes to life.
- This particular method holds the same name as the class and it initializes the object whenever that object is created.
- When you create a class, if you do not create a constructor, the compiler creates one for you; this is useful because it lets all other objects of the program know that the object exists.
- This compiler-created constructor is called the default constructor.
- If you want, you can create your own constructor.

- To create a constructor, declare a method that holds the same name as the class. The method must not return any value.

```
class Flower {
Flower() {
}
}
```

- When you declare an instance of the class, whether you use that object or not, a constructor for the object is created.
- When an instance of a class has been declared, the default constructor is called, whether the object is used or not.


```
Package exercise;
class Flower {
	int type;
	int color;
	char arrangement;
	double unitPrice;
	Flower()
	{
	System.out.println("New Flower
	Order");
	 }
}

public class Exercise {
	static void main(String[] args) {
		Flower flr = new Flower();
	}
}
```


lecture0515

Check lab manual

Nested classes

- A class can be created inside of another class. A class created inside of another is referred to as nested.
- To nest a class, simply create it as you would any other.
- Here is an example of a class called Inside that is nested in a class called Outside:

```
public class Outside {
public class Inside {
}
}
```


- In the same way, you can nest as many classes as you wish in another class and you can nest as many classes inside of other nested classes if you judge it necessary.

```
class Outside {

   class Inside {

      Inside() {
         System.out.println(" -= Inside =-");
      }
   }

   Outside() {
      System.out.println(" =- Outside -=");
      Inside is=new Inside();
   }
}
```


lecture0516

Check lab manual

An Object as a Field in class

- class as a member variable of another class, simply declare its variable as you would proceed with any of the member variables we have declared so far.

```
class Point {
int x;
int y;
}
public class CoordinateSystem {
Point ptStart;
}
```

lecture0517

Check lab manual

Returning an Object From a Method

- Like a value from a regular type, you can return a class value from a method of a class.
- To do this, you can first declare the method and specify the class as the return type

```
class Point {
	int x;
	int y;
	}
public class CoordinateSystem {
	Point ptStart;
	Point ptEnd;
	Point getThePoint() {
	}
}
```


- After implementing the method, you must return a value that is conform to the class, otherwise you would receive an error when compiling the application.

```
class Point {
	int x;
	int y;
	}
public class CoordinateSystem {
	Point ptStart;
	Point ptEnd;
	Point getThePoint(int c) {
	If(c==1) return ptStart;
	Else if(c==2) return ptEnd;
	}
}
```


lecture0518

Check lab manual

lecture0519

Check lab manual

Passing an Object as Argument

- Once a class has been created, it can be used like any other variable.
- For example, its variable can be passed as argument to a method of another class.
- When a class is passed as argument, its members are available to the method that uses it.

lecture0520

Check lab manual

lecture0521

Check lab manual

Passing a Class as its Own Argument

- An instance of a class can be passed as an argument to one of its own methods
- To do this, you primarily pass the argument as if it were any class

```
class Point {
	int x;
	int y;
	void Copy(Point same) {
		this.x = same.x;
		this.y = same.y;
	}
}
```


lecture0522

Check lab manual

- Instead of first declaring a variable of the class and initializing it, you can create an instance of the class in the parentheses of the calling method.
- To do this, you may need a constructor that can specify the values of the fields of the class so the argument can be rightfully initialized.

```
class Point {
	int x;
	int y;
	Point (int x1,int y1){
	this.x=x1; this.y=y1;
	}
	void Copy(Point same) {
		this.x = same.x;
		this.y = same.y;
	}
```



```
}
static void main(String[] args) {
Point p = new Point();
p.copy(new Point(2,3));
}
```

lecture0523

Check lab manual

lecture0524

Check lab manual

Returning a Class From its Own Method

- You can create a method in a class that returns an instance of the class.
- To start, on the left side of the method,

```
class Point {
    Point MethodName() {
    }
}
```

- There are various ways you can deal with the method. If you want to return a new value of the class, you can declare an instance of the class, initialize it, and then return it
- Remember that, to call the method, if it is not static, you will need to declare an instance of the class from where you are calling the method.

lecture0525

Check lab manual

Thanks,

See you next Lecture, isA