CS101 Lecture Notes
March 14, 16 2007
Prepared by Mehmet Can Eksi, Canberk Erol, Oguz Kaysi, M. Cem Olcay

Introduction to Class Writing
(Does Not Reflect Class Discussion)

Example Class

```
public class Flight {

   //variables
   private String name;
   private int number;
   private String origin;
   private String destination;

   //constructor
   public Flight(String airlineName,int flightNumber,String flightOrigin,String
destinationCity )
   {
         name=airlineName;
         number=flightNumber;
         origin=flightOrigin;
         destination=destinationCity;
   }
   //methods

   //setter and getter for airline name
   public void setName (String name1){
         name=name1;
   }
   public String getName(){
         return name;
   }

   //setter and getter for flightnumber
   public void setFlightNumber(int value){
         number=value;
   }
   public int getFlightNumber( ){
         return number;
```

```
    }

    //setter and getter for origin
    public void setOrigin(String place){
        origin=place;
    }
    public String getOrigin(){
        return origin;
    }

    //setter and getter for destination
    public void setDestination(String place1){
        destination=place1;
    }
    public String getDestination(){
        return destination;
    }

    //toString method for returning one line description of the flight
    public String toString(){
        return name+","+(number)+" :from "+(origin)+" to "+(destination);
    }
}
```

Test Class

```
//some exercises on flight class

public class FlightTest {

    public static void main(String[]args){

        //creating two flight objects

        Flight flight1=new Flight("THY",63,"ANKARA","IZMIR");
        Flight flight2=new Flight("AL ITALIA",3445,"ISTANBUL","ROMA");

        //print the flights with the written informations
        System.out.println(flight1);
        System.out.println(flight2);

        //changing the airlinename of flight1 and changing flightnumber of flight2
        flight1.setName("ONUR AIR");
        flight2.setFlightNumber(1888);
```

```
        System.out.println(flight1);
        System.out.println(flight2);

        //just print their destinations
        System.out.println(flight1.getDestination());
        System.out.println(flight2.getDestination());

        //changing flight1's all properties
        flight1.setName("PEGASUS");
        flight1.setFlightNumber(1453);
        flight1.setOrigin("HAKKARİ");
        flight1.setDestination("URFA");

        //just print name of new properties of flight1
        System.out.println(flight1.getName());

        //print all new properties of flight1
        System.out.println(flight1.getName()+" "+flight1.getFlightNumber()+"
"+flight1.getOrigin()
        +" "+flight1.getDestination());

        //or just print the object(it will use also toString method)
        System.out.println(flight1);
    }
    }
```

Questions about Classes

QUESTIONS

1. Write a class which makes Coffee. We need to set its sugar, cream and coffee amount. Then we need to mix them and show us how much coffee, how much sugar, how much cream in it.

Use a constructor with 3 variables in it. Use setcoffeeAmount, setsugarAmount, setcreamAmount and getcoffeeAmount, getsugarAmount, getcreamAmount and mix method and also toString method.

THE OUTPUT SHOULD BE LIKE THAT:

Coffee 1 includes 5.0 gr coffee 3.0 gr sugar 7.0 gr cream
Coffee 1 includes 8.0 gr coffee 2.0 gr sugar 10.0 gr cream

Coffee 1 includes 6.0 gr coffee 3.0 gr sugar 7.0 gr cream
Coffee 1 includes 8.0 gr coffee 2.0 gr sugar 9.0 gr cream

The mix of two coffee includes 14.0 gr coffee 5.0 gr sugar 16.0 gr ceam

2. What is the difference between global variable and local variable?

3. Why do the most objects have accessor or mutator methods?

4. What are the things which seperate the two different methods? What is their signature ?

5. What do the variables represent ? What do the methods represent?

6. In classes, why do we use private variables instead of public variables ?

7. Why do we define the listenners as inner classes?

8. Why do not the constructors have any return type?

9. Which steps do we use for writing a GUI ?

10. Write complex number(3 + 5i) class which real and imaginary part in its constructor. Use setImaginaryPart, setRealPart, getImaginaryPart,

getRealPart methods. And also use addOnto , Complex Conjugate, Complex add and toString.

THE OUTPUT SHOULD BE LIKE THAT:

Complex 1: 10.0 + 20.0i
Complex 2: 15.0 + 3.0i

Complex 1: 9.0 + 20.0i
Complex 2: 15.0 + 7.0i

Addition: 24.0 + 27.0i

Complex: 24.0 + 27.0i

The Conjuge of Complex: 24.0 + -27.0i

ANSWERS

1.

```
public class Coffee
{

	private double coffeeAmount,sugarAmount,creamAmount;

	public Coffee(double coffee,double sugar,double cream)
	{
		coffeeAmount=coffee;
		sugarAmount=sugar;
		creamAmount=cream;
	}
	public void setcoffeeAmount(double amount1)
	{
		coffeeAmount=amount1;
	}
	public void setsugarAmount(double amount2)
	{
		sugarAmount=amount2;
	}
	public void setcreamAmount(double amount3)
```

```
	{
		creamAmount=amount3;
	}

	public double getCoffeeAmount()
	{
		return coffeeAmount;
	}
	public double getsugarAmount()
	{
		return sugarAmount;
	}
	public double getcreamAmount()
	{
		return creamAmount;
	}

	public Coffee mix(Coffee other)
	{
		double newcoffeeAmount=coffeeAmount+other.coffeeAmount;
		double newsugarAmount=sugarAmount+other.sugarAmount;
		double newcreamAmount=creamAmount+other.creamAmount;
		Coffee mixture=new
Coffee(newcoffeeAmount,newsugarAmount,newcreamAmount);
		return mixture;
	}

	public String toString()
	{
		String cof=Double.toString(coffeeAmount);
		String sug=Double.toString(sugarAmount);
		String cre=Double.toString(creamAmount);
		String result=(cof+" gr Coffee, "+sug+" gr Sugar, "+cre+" gr
Cream");
		return result;
	}

}
```

```
public class CoffeeDriver
{
	public static void main(String[]args)
	{
		Coffee Coffee_1=new Coffee(5,3,7);
		Coffee Coffee_2=new Coffee(8,2,10);
		System.out.println("Coffee 1 includes "+Coffee_1);
		System.out.println("Coffee 2 includes "+Coffee_2);

		Coffee_1.setcoffeeAmount(6);
		Coffee_2.setcreamAmount(9);
		System.out.println("\nCoffee 1 includes "+Coffee_1);
		System.out.println("Coffee 2 includes "+Coffee_2);

		Coffee Coffee_Mixture=new Coffee(1,1,1);
		Coffee_Mixture=Coffee_1.mix(Coffee_2);
		System.out.println("\nThe mixture of two Coffee includes
"+Coffee_Mixture);


	}
}
```

2. Local variables are removed when their scope ends but global variables are removed when the program ends.

3. Because accessors and mutator methods allow the client to manage data in controlled manner.

4. Number of the parameters and their order and methods names. The return type of the method does not change it.

5. Variables define the state of the object and methods define its behaviour because we use methods to reach objects and we only make some interreaction throughout methods we can change objects behaviour by methods

6. Because public variables allow code external to the class in which the data is defined to reach in and Access or modify the value of the data. Therefore instance data should be defined with private visibility. Data that is declared as private can be accessed only by the methods of the class. Instance variables should be declared with private visibility to promote encapsulation

7. Listeners are often defined as inner classes because of the intimate relationship between the listener and the GUI components.

8. Because they do not behave exactly as a method. They just allocate the memory. They do not need to send any variable.

9. For creating a Java program that uses a GUI, we must:
    a. Instantiate and set up the necessary components.
    b. Implement listener classes that define what happens when particular events occur.
    c. Establish the relationship between the listeners and the components that generate the events of interest.

10.

```
public class Complex
{
	private double a,b;

	public Complex(double RealPart,double ImaginaryPart)
	{
		a=RealPart;
		b=ImaginaryPart;
	}

	public void setRealPart(double RPart)
	{
		a=RPart;
	}
	public void setImaginaryPart(double IPart)
	{
		b=IPart;
```

```
	}

	public double getRealPart()
	{
		return a;
	}
	public double getImaginaryPart()
	{
		return b;
	}
	public void addOnto(Complex other)
	{
		a=a+other.a;
		b=b+other.b;
	}

	public Complex Conjugate()
	{
		double na=a,nb=-b;
		Complex newCon=new Complex(na,nb);
		return newCon;
	}
	public Complex add(Complex other)
	{
		double newa=a+other.a;
		double newb=b+other.b;
		Complex newSum=new Complex(newa,newb);
		return newSum;
	}
	public String toString()
	{
		String result;
		result=Double.toString(a)+" + "+Double.toString(b)+"i";
		return result;
	}

	}



public class ComplexDriver
{
	public static void main(String[]args)
```

```
	{
		Complex Complex_1=new Complex(10,20);
		Complex Complex_2=new Complex(15,3);
		System.out.println("Complex 1: "+Complex_1);
		System.out.println("Complex 2: "+Complex_2);
		Complex_1.setRealPart(9);
		Complex_2.setImaginaryPart(7);
		System.out.println("\nComplex 1: "+Complex_1);
		System.out.println("Complex 2: "+Complex_2);

		Complex Addition=Complex_1.add(Complex_2);
		System.out.println("\nAddition: "+Addition);

		Complex_1.addOnto(Complex_2);
		System.out.println("\nComplex 1: "+Complex_1);

		System.out.println("\nThe Conjuge of Complex 1: "+
Complex_1.Conjugate());
	}
}
```