King Fahd University of Petroleum and Minerals
INFORMATION & COMPUTER SCIENCE DEPARTMENT, KFUPM
ICS 102 SECTION 54 & 57 (002 Spring-Semester)
INTRODUCTION TO COMPUTING
LAB #5 Classes an Objects

Instructor: Mustafa A. Abuosba

Lab Objectives

- Implementing simple classes
- Understanding methods and instance variables
- Object construction
- Methods

When you think in an object-oriented manner, everything is an object, and every object is a member of a class.

Creating a Class

When you create a class, first you must assign a name to the class, and then you must determine what data and methods will be part of the class. Suppose you decide to create a class named Employee. One instance variable of Employee might be an employee number, and two necessary methods might be a method to set or provide a value for the employee number and another method

Class Definition

To program in Java the main thing you do is write class definitions for the various objects that will make up the program. A class definition encapsu
been defined, it serves as a template, or blueprint, for creating individual objects or instances of the class.

A class definition contains two types of elements: variables and methods. Variables are used

you need to answer five basic questions

1- What tasks will the object perform?
2- What information will it need to perform its tasks?
3- What methods will it use to process its information?
4- What information will it make public for other objects?
5- What information will it hide from other objects?

Constructors The purpose of a constructor is to initialize the instance variables of an object Constructors always have the same name as their class. Constructors are generally declared as public to enable any code in program to construct new objects of the class, If you do not initialize an instance variable that is a number, it is initialized automatically to zero

example

```
public Rectangle(double l, double w){ // Constructor method
                length=l;
                width=w;
        }
```

Instance variables
Instance variables are generally declared with the access specifier private. That means, they can be accessed only by the methods of the same class, not by any other method.
Example

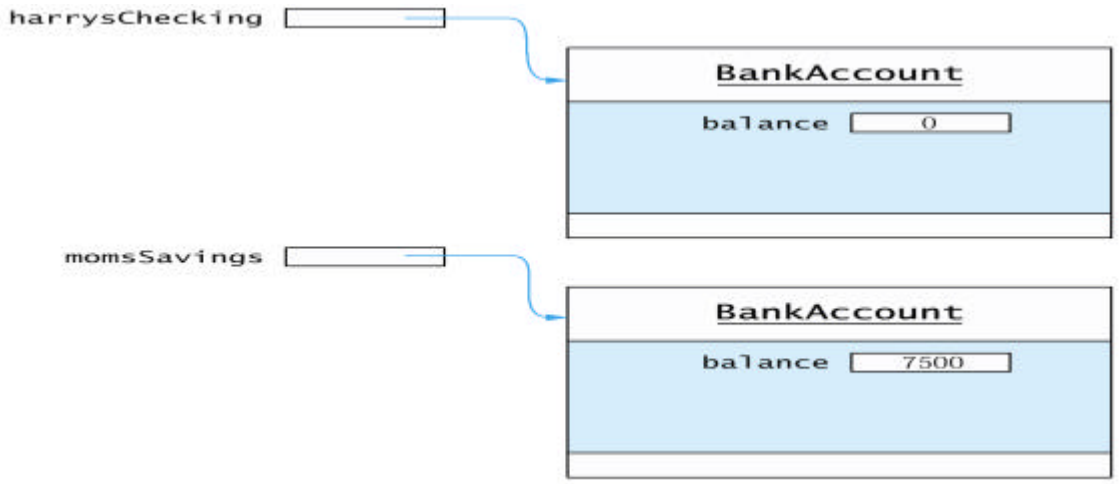
public class BankAccount{

private double balance;

}

BankAccount harrysChecking=new BankAccount();

BankAccount momsSavings= new BankAccount();l



HarrysChecking and momsSavings are two objects of BankAcount each object have its own banlace field

---

Private:
instance variables declared as private so that they can be accessed only by methods of the same class, like deposit( ), withdraw( ), getBalance( ).

Signature:

which is comprised of the name of the method and the types and order of the parameters in the parameter list

Several methods can have the same name, as long as their signatures are different.
Methods within a class must have different signatures. Two methods with the same name must have different numbers of parameters, or if both have the same number of parameters, not all corresponding parameters can have the same type

---

Example the Rectangle class definition

```
public class Rectangle{
        private double length; // instance variables
        private double width;
        public Rectangle(double l, double w){ // Constructor method
                length=l;
                width=w;
        }

        public double calculateArea( ) // access method
        {
                return length *width;

        }
}// Rectangle class
```

The following create, or instantiate, two instances of the Rectangle class

```
Rectangle rectangle1=new Rectangle(30,10);
Rectangle rectangle2=new Rectangle(30,10);
```

The rectangle1 object is an instance of the Rectangle class. In this statement it is given 30 and 10 as the initial values of its length and width.
Once we have created Rectangle instances and given them their initial length and width, we can ask each rectangle to calculate and tell us its area

```
Rectangle1.calculateArea()
Rectangle2.calculateArea()
```

Define, Create, Use

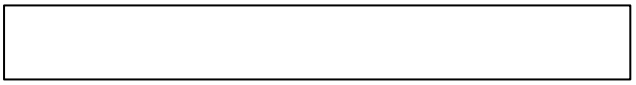
steps

1- Define one or more classes(class definition)
2- Create objects as instances of the classes
3- Use the objects to do tasks

The Rectangle class is not a complete Java application program, because it does not have a main( )

```
public class RectangleUser{
        public static void main(String args[]){
                Rectangle rectangle1=new Rectangle(30,10);
                Rectangle rectangle2=new Rectangle(25,20);
                System.out.println("rectangle1 area= "+rectangle1.calculateArea());
                System.out.println("rectangle2 area= "+rectangle2.calculateArea());
        }
}
```

What was the area of rectangle1 and rectangle2 ?



Access Methods:

Access methods fall into two categories:

Accessor(or getters) methods that return the value of a private variable

Mutators (or setters) methods that change the value of a private variable

Mutators should have a return type of void. This not a rule of the Java language, but just a recommendation to make it easy to differentiate between mutators and accessors . example

```
public class SS{
        private int x1;


        public SS (int x){//constructor
                x1=x;
        }

        // setter method to change the values of x1
        public  void setNum(int n){
                x1 =n;}



        public static void main(String[] args){

        SS s1=new SS(44);

        //s1.x1=55; wrong

        s1.setNum(100); // to change the value

        System.out.println(s1.x1);
}
}
```

Exercises#1 Study the following program

```
public class Account{
        //instance variables
        private double balance;
        //constructors
        public Account(double initialBalance){
                balance=initialBalance;
        }
        public Account(){
                balance=0.0;
        }
        // instance methods
        public void deposit(double amount){
                balance =balance + amount;
        }
        public void withdraw(double amount){
                balance= balance- amount;
        }
        public double getBalance(){
                return balance;
        }
        public  void close(){
                balance=0.0;
         }
}
```

```
// The following program will test the Account class

public class TestAccount{
        public static void main(String[] args){
                Account acct1=new Account(1000.0);
                System.out.println("Balance in account1= "+acct1.getBalance());
                acct1.deposit(100.00);
                System.out.println("Balance in account1= "+acct1.getBalance());
                acct1.withdraw(150.00);
                System.out.println("Balance in account1= "+acct1.getBalance());
                acct1.close();
                System.out.println("Balance in account1= "+acct1.getBalance());
                Account  acct2=new Account();
                acct2.deposit(500.00);
                System.out.println("Balance in account2= "+acct2.getBalance());
                acct2.withdraw(350.00);
                System.out.println("Balance in account2= "+acct2.getBalance());
                acct2.close();
                System.out.println("Balance in account2= "+acct2.getBalance());
        }
}
```

Exercise#2

Given the definition of NumberAdder classs, add statements to its main ( ) method to create two instances of this class, named **adder1**, **adder2,**
and 15, And to
sums . use setNums to change the values of adder1 by adding 20 and 40 to the original numbers

```
public class NumberAdder{
 private double num1;
 private double num2;
 public void setNums(double n1, double n2){//setter method
  //add statement here
 // add statement here
 }

 public double getSum( ){
  return num1 + num2;
 }

 public static void main(String [ ] args){

// add statements here

 }

}
```

Exercise#3

Write a class definition for **Square**, declare two reference variable square1(20) and square2(15) . test this class by printing the result of computing the perimeter of square and area of square object that is capable of re with two methods, calcPermeter() ➔ 4 * side
calcArea ( ) ➔ side * side

Exercise#4

Implement a class **Employee** with the following fields, constructors and methods:

Fields: name, salary;
Constructors:
public Employee(String name, double salary)
public Employee(double salary, String name)
public Employee(String name) //this should initialize the salary to 0.0
Methods: public String getName()
public double getSalary()
public void raiseSalary(double percentageIncrease)
s name and Salary
public String toString( )Write an application **TestEmployee** that reads the names and salaries of two employees and make use of the **Employee** class to create two Employee objects and print them. Next, read a percentage increase in salary for each employee and call the ***raiseSalary()*** method to increment each and print the objects again.