

Department of Automation & Computer-Aided Engineering**ACE 1050 Design Computing (Fall 2007)****Assignment #4 Suggested Solution****Question 1**

Instance variable is a data item owned by a particular instance of a class. Each instance of the class has its own unique copy of the variable, which is allocated when the object is allocated via new. It is a symbolic name for a data field, usually has private access to assist in information hiding by encapsulating the object's attributes.

Static methods may not access the instance variables of their class (or any other class for that matter), other than via some object reference. They may even access private instance variables in their class via some object reference.

Instance method is a non-static method that can only be invoked on an instance of a class. Methods are instance methods unless they contain the static keyword in their declaration. When an instance method is being executed, the attributes of the object associated with the invocation are accessed and manipulated.

Instance method is considered as a subroutine or function designed to work on the current object. Methods are always part of some class. An instance method has access to all the instance variables, other instance methods, as well as the static class-as-a-whole methods and variables.

Static methods may not access the instance methods of their class (or any other class for that matter), other than via some object reference. They may even access private instance methods in their class via some object reference.

Constructor is a special type of instance method that creates a new object. In Java, constructors have the same name as their class and have no return value in their declaration. There is more than one kind of construction:

- **Default construction** constructs a class representation using default values for its attributes. No input parameters are required.
- **Specific construction** constructs a class representation using supplied values for its attributes. Input parameters are required.

Accessors supply the values of the attributes. It is an individual method for providing the value of class variables. Each accessor execution produces a return value without input parameters, and the return value is the value of the invocation. It is a method used to examine the members of a class. Variables declared as private are accessed indirectly via these methods.

Mutators manage requests for changing attributes to ensure the objects always have sensible values. It is an individual method for setting the value of class variables. It is readable, less error prone, and facilitates enhancements through localization. It is a method used to modify the members of a class. Variables declared as private are accessed indirectly via these methods.

Question 2

```
//// class Account code start
```

```
public class Account {  
    // instance variables  
    private String name;  
    private String owner;  
    private double balance;  
    // Default constructor: a new account with name "name", "owner", and balance 0.  
    public Account() {  
        name = "name";  
        owner = "owner";  
        balance = 0;  
    }  
    // Specific constructor  
    public Account(String s, String t, double n) {  
        setName(s);  
        setOwner(t);  
        balance = n;  
    }  
    // Mutators  
    // void setName (String s): sets the name of the associated account to s.  
    public void setName(String s) {  
        name = s;  
    }  
    // void setOwner (String s): sets the owner of the associated account to s.  
    public void setOwner(String s) {  
        owner = s;  
    }  
    // void deposit (double n): adds amount n to current balance of associated account.  
    public void deposit(double n) {  
        balance += n;  
    }  
    // void withdraw (double n): subtracts n to current balance of associated account.  
    public void withdraw(double n) {  
        if (balance - n < 0) {  
            System.out.println("Insufficient balance, withdraw fail!");  
        }  
        else {  
            balance -= n;  
        }  
    }  
    // void close (): zeros out the account balance, set the owner to " ".  
    public void close() {  
        balance = 0;  
        setOwner(" ");  
    }  
}
```

```

// Accessors
// String getAccountName (): returns the name of the associated account.
public String getAccountName() {
    return name;
}
// String getOwner (): returns the owner of the associated account.
public String getOwner() {
    return owner;
}
// double getBalance (): returns the balance of the associated account to s.
public double getBalance() {
    return balance;
}
// String toString (): returns a textual output of attributes of associated account.
public String toString() {
    String printoutput;
    printoutput = "Account Name: "+ name + "\n";
    printoutput += "Account Owner: "+ owner + "\n";
    printoutput += "Account Balance: " + Math.round(balance);
    return printoutput;
}
}
//// class Account code end

// Q4.24 demo main program code start
/*
i. Set the initial account name as your student ID, account owner as you.
ii. Deposit 100
iii. Withdraw 50
iv. Change owner to "Yao Zhiyang"
v. Printout the attributes of the final account information using the toString()
method in the following format:
Account Name: 26098045
Account Owner: Yao Zhiyang
*/
import java.io.*;
public class hw4q2 {
    public static void main(String[] args) {
        Account a = new Account("05141580","Ko Pui Hang", 0);
        a.deposit(100);
        a.withdraw(50);
        a.setOwner("Yao Zhiyang");
        System.out.println(a.toString());
    }
}
// demo main program code end

```

```

C:\Program Files\Xinox Software\JCre
Account Name: 05141580
Account Owner: Yao Zhiyang
Account Balance: 50
Press any key to continue...

```

Question 3

```

//// class Date code start

```

```

public class Date {
    // instance variables
    private int day;
    private int month;
    private int year;
    // Default constructor: configures a new Date to represent the date 1 January 2000.
    public Date() {
        day = 1;
        month = 1;
        year = 2000;
    }
    // Specific constructor
    public Date(int in_d, int in_m, int in_y) {
        setDay(in_d);
        setMonth(in_m);
        setYear(in_y);
    }
    // Mutators
    // void setDay (int in_d): sets the day of the associated Date to in_d.
    public void setDay(int in_d) {
        day = in_d;
    }
    // void setMonth (int in_m): sets the month of the associated Month to in_m.
    public void setMonth(int in_m) {
        month = in_m;
    }
    // void setYear (int in_y): sets the year of the associated Year to in_y.
    public void setYear(int in_y) {
        year = in_y;
    }
    // Accessors
    // int getDay (): returns the day of the associated Date.
    public int getDay() {
        return day;
    }
    // int getMonth (): returns the month of the associated Date.
    public int getMonth() {
        return month;
    }
}

```

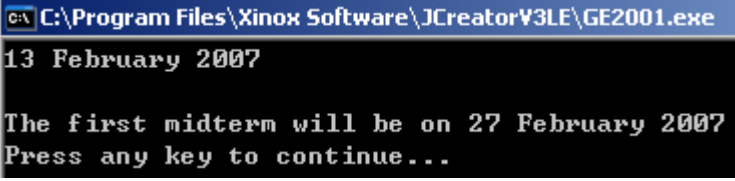
```

    // int getYear (): returns the year of the associated Date.
    public int getYear() {
        return year;
    }
    // String toString (): returns a textual output of the attributes of the associated Date.
    public String toString() {
        String m_string[] = {
            "Zero", "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "Novemeber",
"December"
        };
        String printoutput;
        printoutput = day + " ";
        printoutput += m_string[month] + " ";
        printoutput += year;
        return printoutput;
    }
}
//// class Date code end

// Q4.25 main program to demo Date class
/* 1. Configure a new date "13 February 2007", output it.
 * 2. Change the date to "27 February 2007" using the setDay method,
 * and output the following message:
 * "The first midterm will be on 27 February 2007",
 * where the value of "27", "February" and "2007",
 * should be obtained by getDay, getMonth and getYear methods.
 */
import java.io.*;
public class hw4q3 {
    public static void main(String[] args) {
        // output 1st date
        Date dmy = new Date(13, 2, 2007);
        System.out.println(dmy.toString() + "\n");
        // get 2nd date parameters
        int new_d = dmy.getDay() + 14;
        String m_string[] = {
            "Zero", "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "Novemeber",
"December"
        };
        String new_m = m_string[dmy.getMonth()];
        int new_y = dmy.getYear();
        // output 2nd date
        System.out.print("The first midterm will be on " + new_d);
        System.out.print(" " + new_m + " " + new_y + "\n");
    }
}

```

```
    }  
}  
// demo main program code end
```



```
C:\Program Files\Xinox Software\JCreatorV3LE\GE2001.exe  
13 February 2007  
  
The first midterm will be on 27 February 2007  
Press any key to continue...
```