Lab2

March 6, 2024

1 Lab 2

Deadline: **Week 3** in your respective lab session

1.0.1 Name:Hansel Rodrigues

1.0.2 Student ID: 230603866

1.1 Question 1 [1 mark]

Write a class `BankAccount`, with instance variables `accountNumber` and `balance`. The `balance` by default should be set to `0`. Apart from appropriate accessor methods and a constructor, you need to implement two instance methods: `deposit` and `withdraw`.

`deposit` - should take any number as an argument, check if it is a valid number (greater than `0`), and add it to the `balance` if it is a valid number. If successful, return `true`; otherwise, return `false`.

`withdraw` - should take any number as an argument, check if it is a valid number (greater than `0` and less than the `balance`) and deduct it from the balance if it is a valid number. If successful, return `true`; otherwise, return `false`.

Lastly define class `Main` with the main method to test your code.

The main objective of this exercise is to use an appropriate access modifier to encapsulate the data.

Write your answer below:

[1]:
```
public class BankAccount
{
    String accountNumber;
    double balance;

    public BankAccount (String accountNumber)
    {
        this.accountNumber = accountNumber;
        this.balance = 0;
    }

    public String getAccountNumber ()
```

```
    {
        return accountNumber;
    }
    public double getBalance()
    {
        return balance;
    }

    public boolean deposit (double amount)
    {
        if (amount > 0)
        {
            balance = balance + amount;
            return true;
        }
        else
        {
            return false;
        }
    }

    public boolean withdraw (double amount)
    {
        if (amount > 0 && amount <=balance)
        {
            balance = balance - amount;
            return true;
        }
        else
        {
            return false;
        }
    }

}
```

[2]:
```
public class Main1 {
    public static void main(String[] args) {
        BankAccount ba = new BankAccount("123456789");

        System.out.println(ba.getAccountNumber());
        System.out.println(ba.getBalance());

        double depositAmount = 100;
        System.out.println(ba.deposit(depositAmount));        // true

        double invalidDepositAmount = -100;
```

```
        System.out.println(ba.deposit(invalidDepositAmount)); // false

        double withdrawAmount = 50;
        System.out.println(ba.withdraw(withdrawAmount));      // true

        double tooHighWithdrawal = 200;
        System.out.println(ba.withdraw(tooHighWithdrawal));   // false

        double tooLowWithdrawal = -200;
        System.out.println(ba.withdraw(tooLowWithdrawal));    // false

        System.out.println(ba.getAccountNumber());
        System.out.println(ba.getBalance());
    }
}
```

Run your program:

[3]:
```
Main1.main(null)
```

```
123456789
0.0
true
false
true
false
false
123456789
50.0
```

1.2 Question 2 [1 mark]

Write a class `Student` with 2 instance variables, `name` and `id`. It also contains a constructor which initialises `name` and `id` to the values passed as an argument.

Implement a class method (i.e. a static method) `checkDuplicates` inside the `Student` class, which takes an array of `Student` elements as an argument and checks whether there are two identical students in the array. If yes, it should return `true` and `false` otherwise.

Lastly, define class `Main2` with the main method and test your code. Test at least one array with a duplicate and one without duplicates.

Write your answer below:

[4]:
```
class Student
{
    String name;
    int id;
```

```
    public Student (String name, int id)
    {
        this.name = name;
        this.id = id;
    }
    static boolean checkDuplicates(Student[] s)
    {
        for (int i=0; i<s.length-1; i++)
        {
            for (int j=i+1;j<s.length; j++)
            {
                if (s[i].name.equals(s[j].name) && s[i].id == s[j].id)
                {
                    return true;
                }

            }
        }
        return false;
    }
}
```

[6]:
```
public class Main2 {
    public static void main(String[] args) {

        Student[] studentsArrayWithDuplicate = {
                new Student("Alice", 1),
                new Student("Bob", 2),
                new Student("Charlie", 3),
                new Student("Alice", 1)
        };

        Student[] studentsArrayWithoutDuplicate = {
                new Student("Alice", 1),
                new Student("Bob", 2),
                new Student("Charlie", 3),
                new Student("David", 4)
        };

        System.out.println(Student.checkDuplicates(studentsArrayWithDuplicate));
        System.out.println(Student.
 ↪checkDuplicates(studentsArrayWithoutDuplicate));
    }
}
```

Run your program:

[7]:
```
Main2.main(null);
```

```
true
false
```

1.3 Question 3 [1 mark]

Write a method `sortStudents` which, given an array of `Student` elements, sorts it using the Bubble Sort algorithm by `name` in alphabetical order. If more than one student has the same `name`, sort it by `id` in ascending order. You can assume that each `id` is unique.

Test your code!

Write your answer below:

[8]:
```
class Student
{
    String name;
    int id;

    public Student (String name, int id)
    {
        this.name = name;
        this.id = id;
    }
}
public static void sortStudents(Student[] s)
{
    int length = s.length;
    for (int i=0; i<length-1; i++)
    {
        for (int j=0; j<length-i-1; j++)
        {
            int sortingNames = s[j].name.compareTo(s[j+1].name);
            if (sortingNames > 0 || (sortingNames == 0 && s[j].id > s[j+1].id))
            {
                Student temp = s[j];
                s[j] = s[j+1];
                s[j+1] = temp;
            }
        }
    }

}
```

Run your program:

[9]:
```
Student[] students = {
    new Student("John", 3),
    new Student("Alice", 2),
    new Student("Bob", 1),
    new Student("Bob", 5)
};

sortStudents(students);

for (Student s : students)
    System.out.println("(" + s.name + "," + Integer.toString(s.id) + ")" );
```

```
(Alice,2)
(Bob,1)
(Bob,5)
(John,3)
```

1.4 Question 4 [1 mark]

Notice that for the `Student` class of the previous two questions, you can create two objects `s1` and `s2` which have identical `id` and `name`. In this exercise we will modify the `Student` class to make creating such two such objects impossible. It is for this reason that the constructor of the modified class `Student4` below is set to `private`. This makes it impossible to create objects of this class from outisde. Objects instead will be created by the static method `register`.

Modify the static method `register` to check whether a student with this `name` and `id` was registered before. If yes, return a reference to the previously created instance of a `Student4`; if not, create a new instance of `Student4` using passed values and return its reference. You are allowed to modify the `Student4` class to achieve this.

Define the `Main4` class to test your code. You should check whether the `register` function is returning the correct reference and whether it prints out the names of all registered students.

You can assume that the maximum number of registered students does not exceed `30`.

HINT: Keep track of the instances that have been created before by using a static array of type `Student4`.

Write your answer below:

[41]:
```
class Student4 {
    String name;
    int id;
    static final int MAX_REGISTERED = 30;
    static int numberOfStudents = 0;
    static Student4[] registeredStudents = new Student4[MAX_REGISTERED];

    private Student4(String name, int id) {
        this.name = name;
```

```
        this.id = id;
    }

    public static Student4 register(String name, int id)
    {
        for (int i=0; i<numberOfStudents; i++)
        {
            if (registeredStudents[i].name.equals(name) &&␣
 ↪registeredStudents[i].id == id)
            {
                return registeredStudents[i];
            }
        }

        Student4 newStudent = new Student4(name,id);
        registeredStudents[numberOfStudents++] = newStudent;
        return newStudent;

    }
}
```

[42]:
```
class Main4 {
    public static void main(String[] args) {
        Student4 student1 = Student4.register("John", 123);
        Student4 student2 = Student4.register("Jane", 456);
        Student4 student3 = Student4.register("John", 123);

        System.out.println(student1 == student3); // true
        System.out.println(student1 == student2); // false

        for (int i = 0; i < Student4.numberOfStudents; i++)
            System.out.println(Student4.registeredStudents[i].name + " " +␣
 ↪Student4.registeredStudents[i].id);
    }
}
```

[43]:
```
Main4.main(null);
```

```
true
false
John 123
Jane 456
```

1.5 Question 5 [1 mark]

Consider the class `Employee` below, it has two instance variables `name` of type `String` and `manager` of type `Employee`.

A *district manager* is an employee that does not have any manager, i.e. its `manager` instance variable is set to `null`. Write the method `getDistrictManager()` which returns the district manager of the given employee.

In other words if we have employees `jane`, `joe`, and `john`, such that `jane` is the manager of `joe` and `joe` is the manager of `john`, and moreover `jane` does not have a manager; then calling `john.getDistrictManager()` should return an object reference to `jane`.

Finally test your code with the example above.

Write your answer below:

[47]:
```
class Employee {
    String name;
    Employee manager;

    Employee(String name, Employee manager) {
        this.name = name;
        this.manager = manager;
    }

    public Employee getDistrictManager() {
        if (manager == null)
        {
            return this;
        }
        else
        {
            return manager.getDistrictManager();
        }

    }
}
```

[48]:
```
class Main5 {
    public static void main(String[] args) {
        Employee jane = new Employee("jane", null);
        Employee joe = new Employee("joe", jane);
        Employee john = new Employee("john", joe);

        System.out.println(john.getDistrictManager().name); // should print jane
        System.out.println(jane.getDistrictManager().name); // should print jane
    }
}
```

[49]:
```
Main5.main(null);
```

```
jane
jane
```

[ ]:
```

```