Data structure

Lab 3 – class and object - part2

Dr. Sophea PRUM
sopheaprum@gmail.com

****

If a programmer codes just for fun he has all his skill.
If he codes for score his hand tremble and his breath is uneasy

****

Submit your project by email

1. First submission at the end of the session
2. Second submission before Thursday 23rd, 2017 (mid-night)

For these two submissions, export your project as a Zip file and send to sopheaprum@gmail.com

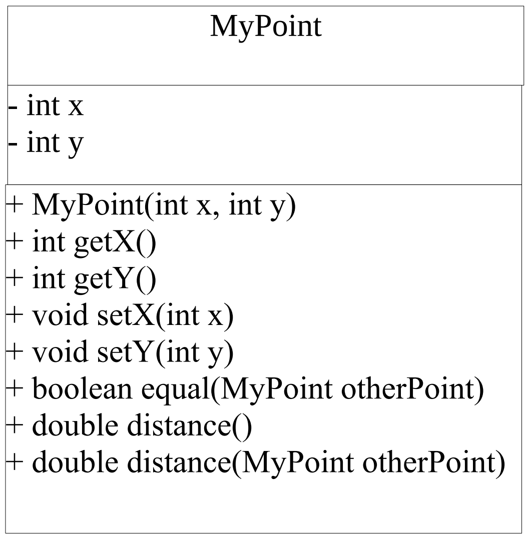
To export your project: File → Export project → To Zip → Export

---

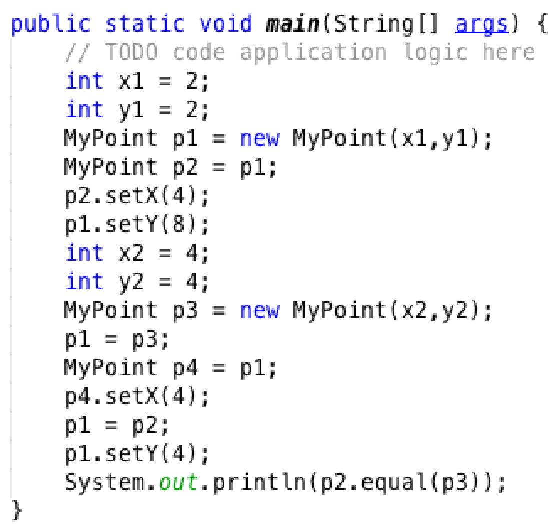
Exercise 1: MyPoint

TODO1: Create a java class and name it *MyPoint* according to the UML digram given bellow.
This class has:

- 2 fields: x and y represent the coordinate of each point
- 1 constructor having 2 parameters x and y
- Getter and setter methods of fields x and y
- 3 Methods
  - equal(MyPoint otherPoint): allows to compare of two points are the same. This method return true of the coordinate of these two points are the same. Otherwise, the method return false.
  - distance(): returns the distance between this point and the point (0,0)
  - distance(MyPoint otherPoint): returns the distance between this point and otherPoint



TODO2: Create a new java main class called *FindMyPoint*. In the main method, write the code bellow.



```java
public static void main(String[] args) {
    // TODO code application logic here
    int x1 = 2;
    int y1 = 2;
    MyPoint p1 = new MyPoint(x1,y1);
    MyPoint p2 = p1;
    p2.setX(4);
    p1.setY(8);
    int x2 = 4;
    int y2 = 4;
    MyPoint p3 = new MyPoint(x2,y2);
    p1 = p3;
    MyPoint p4 = p1;
    p4.setX(4);
    p1 = p2;
    p1.setY(4);
    System.out.println(p2.equal(p3));
}
```

TODO3: Give the schema illustrates JVM memory allocation when running this program.

Stack

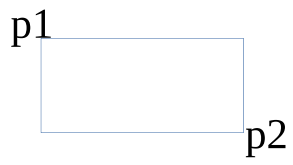
Heap

Exercise 2: MyRectangle

TODO1: Create a new java class and name it *MyRectangle*. Each rectangle is represented by 2 points p1 and p2 representing the two corners of the rectangle, as illustrated in the figure bellow.

This class has:

- 2 fields represent the corners of the rectangle
- 1 constructor taking 2 parameters of p1 and p2 as input
- 3 methods
  - getWidth(): returns width of the rectangle
  - getHeight(): returns height of the rectangle
  - getSurface(): returns surface of the rectangle

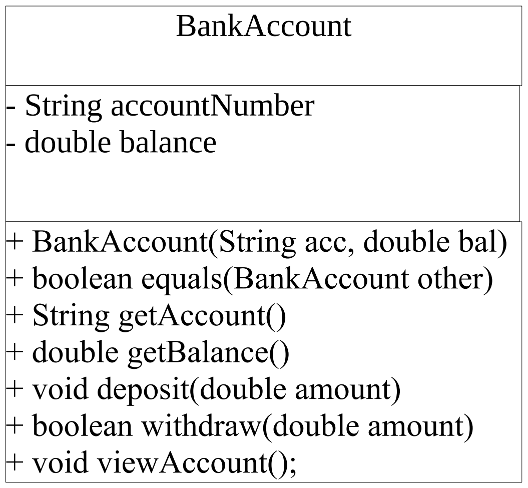


MyRectangle
- MyPoint p1 - MyPoint p2
+ MyRectangle(MyPoint p1, MyPoint P2) + double getWidth() + double getHeight() + double getSurface()

TODO2: Create a java main class called *TestMyRectangle*. In this class, create an instance of class MyRectangle. Print its width, height and surface.

Exercise 3: BankAccount

TODO1: Create a new java class and name it as BankAccount. The UML diagram of this class is illustrated in the figure bellow:


BankAccount
- String accountNumber
- double balance
+ BankAccount(String acc, double bal)
+ boolean equals(BankAccount other)
+ String getAccount()
+ double getBalance()
+ void deposit(double amount)
+ boolean withdraw(double amount)
+ void viewAccount();

This class "BankAccount" has

- 2 fields:
  - accountNumber: an identifier that never changes
  - balance: potentially does change
- 1 constructor
  - Each time an object of class *BankAccount* is created, the values of *accountNumber* and *balance* must be provided. Therefore, this constructor take 2 parameters as input (acc, and bal)
    - acc is account number
    - bal is balance
- 6 methods
  - boolean equals(BankAccount other): a public method having one parameters *other* which represents other object of BankAccount. This method returns true if the accountNumber of the current object is the same as the accountNumber in the object *other*.
  - String getAccount(): a public method which returns the bank account number of this account
  - double getBalance(): a public method which returns the balance of this account
  - void deposit(double amount): deposit an amount in the account. This method returns nothing
  - boolean withdraw(double amount): a public method allowing to withdraw some amount from this account. The withdraw amount must be positive and not zero. Otherwise, the system return false. In addition, the method will refuse the withdraw operation (return false) if the withdraw amount is higher than the balance. If the operation is success, this method return true.
  - void viewAccount(): is public method allowing to view the accountNumber and its balance. This method return nothing. It simply prints out "*Account number xxxxxxx having yyyy$*". Where *xxxxxxxx* must be replaced by the accountNumber and *yyyy* must be replaced by balance.

TODO2: create a new java class called *BankApplication*. This java class contains the main method.

- Q1: Now create an instance called myAccount having accountNumber = “112233445566” with 200$ as the initial balance.
- Q2: View this account
- Q3: You just earn 150$ from your part-time job. Now you deposit all your money and view your account
- Q4: You are planing a trip to Penang. So you withdraw 300$ from your account. The system will print-out “*successfully withdraw yyyy*” if withdraw operation is success, where *yyyy* is the withdraw amount. Otherwise the system will print-out “*withdraw operation has been rejected*”.
- Q5: Now again, you want to buy some books, so you wish to withdraw another 100$. The system will print-out “*successfully withdraw yyyy*” if withdraw operation is success. Otherwise the system will print-out “*withdraw operation has been rejected*”.
- Q6: Create a new instance (bank account) called *otherAccount*. This account having accountNumber = “002233445566” with 500$ as the initial balance. Now, view the otherAccount object
- Q7: We want to check if *myAccount* and *otherAccount* are different or the same. The system will print out “*Duplicate account number*” if this two account are equal. Otherwise, the system will print out “*The two account are different*”.