1. Homework 4

Posted: Septemeber/10/2018
Due: September/23/2018 24.00

All homework solutions are due September/23/2018 24.00. I recommend to submit at least one version of all homework solutions long before due date.

1.1. Homework 4.1 (10 Points)

All homework solutions are due September/23/2018 24.00. I recommend to submit at least one version of all homework solutions long before due date.

Objective: Understanding and Solving a simple Inheritance Problem

Grading:
Correctness: You can lose up to 40% if your solution is not correct
Quality: You can lose up to 80% if your solution is poorly designed
Testing: You can lose up to 50% if your solution is not well tested
Explanation: You can lose up to 100% if your solution if you can not explain your solution during the grading session

Homework Description:

You have to implement the following classes:

- Lion
- Tiger
- Giraffe
- Gazelle

using inheritance, abstract classes, or interfaces as it this seems appropriate.

Explanation:
Use the following information to organize Classes, members, and methods. Include all information: that is relevant to make these objects interact in a general way using inheritance whenever possible to maximize code reuse. Methods can be stubbed out, but the correct method signatures must be present.

There is a zoo, and in that zoo there are 4 kind of animals. There are lions, tigers, giraffes, and gazelles, Some of these animals eat meat and hunt. Some eat plants and they go out to graze. The zookeeper can send them home and the keeper needs to know if they are home. Each animal goes to a :q! kind of home; for example a tiger goes to a den.

You have to implement for all classes above. Keep in mind the following:

- a *Carnivore* is a *Animal*,
- a *Herbivore* is a *Animal*,
- a *Lion* is a *Carnivore*,
- a *Gazelle* is a *Herbivore*,
- etc

Your Work:
You have to use abstract classes, and/or interfaces, and you have to use inheritance. You must be able to explain to your grader your design decisions. Your grades depend on how reasonable your decision are.

Requirements:

You have to name your main class Zoo.java

Example:

An example of a solution execution:

The following is a snipper of a main program, but you have to implement 1 and 2.

```
class Zoo
{
   static Animal[] animals = {
                    new Tiger("Katya")
             };
   public static void printSpecies(Animal thisOne)      {
        System.out.println("I am a " + thisOne.getSpecies());
   }
   public static void printName(Animal thisOne) {
        System.out.println("My name " + thisOne.getName());
   }
   public static void printHomeStatus(Animal thisOne)   {
        System.out.println("I am" + ( thisOne.amIHome() ? " " : " not " ) + "home."  );
        if ( thisOne.amIHome() )
                System.out.println(thisOne.areYouHome());
   }

   public static void main(String args[] )      {
        Class[] classes;
        for ( int index = 0; index < animals.length; index ++ ) {
                printSpecies(animals[index]) ;
                printName(animals[index]) ;
                printHomeStatus(animals[index]) ;
                animals[index].goHome();
                printHomeStatus(animals[index]) ;
                // if I am a tiger send me to hunt and ask if I am hungry - implement 1
                // if I am a gazelle send me to graze and ask if I am hungry - implement 2
        }
   }
}
% java Zoo
I am a tiger
My name Katya
I am not home.
I am home.
I am in my den
```

Submission:

```
% ssh glados.cs.rit.edu # or use queeg.cs.rit.edu if glados is down
# password
# go to the directory where your solution is ...
% try hpb-grd lab4-1 'All files required'
# you can see if your submission was successful:
# try -q hpb-grd lab4-1
```

1.2. Homework 4.2 (10 Points)

Objective: Understanding and Solving a simple Inheritance Problem

Grading:
Correctness: You can lose up to 40% if your solution is not correct
Quality: You can lose up to 80% if your solution is poorly designed
Testing: You can lose up to 50% if your solution is not well tested
Explanation: You can lose up to 100% if your solution if you can not explain your solution during the grading session

Homework Description:

Develop a bank program that can have a checking, savings, and credit card account. Use inheritance to maximize code reuse.

Explanation:

More generally there are different types of accounts,

Asset accounts
: Asset accounts include bank accounts like checking or savings: Deposits are debits. Withdrawals are credits.

Liability accounts
: Loans and credit cards are liability accounts. Received payments (transactions "paying off" your credit card) are debits. Expenses/purchases are credits.

Each account must have a unique account number.

In the example you will see that you need to be able to pass time which will apply interest to each account in the bank. Interest rates must be different for each account type. You must be able to open, close, debit and credit (remember they mean different things for different accounts)

You can make the assumption that input to the program will be valid.

See below for example run of a program where an account is opened and a bank summary is shown for all open accounts. Another account is opened and 3 months pass. You will notice that a credit card bill is sent each month that passes after interest is applied.

Your Work:
The folloing classes might give you an idea how about the used names of interfaces, abstract classes, and classes.

```
% ls *.java
AccountMethods.java  BankAccount.java CreditCardAccount.java   RitBank.java
AssetAccount.java    CheckingAccount.java LiabilityAccount.java    SavingsAccount.java
```

Requirements:
Your system must compile on our systems.

Example:

An example of a solution execution:

```
This is part of an execution of the program:

% java RitBank
Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
```

```
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
open

What type of account
    0 - for savings
    1 - for checking
    2 - for credit card?>
0

What is the customer’s name?>
Bob

How much to deposit?>
300
Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
summary

Bank Summary
Account number: 1594686078
    Account type: SAVINGS
        Customer name: Bob
        Account balance: $300.00

Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
open

What type of account
    0 - for savings
    1 - for checking
    2 - for credit card?>
2

What is the customer’s name?>
```

```
Jack
Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
summary

Bank Summary
Account number: 1594686078
    Account type: SAVINGS
        Customer name: Bob
        Account balance: $300.00
Account number: 144986167
    Account type: CREDIT_CARD
        Customer name: Jack
        Amount owed: $.00

Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
credit

What is the account number?>
144986167

How much?>
550
Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
summary

Bank Summary
Account number: 1594686078
```

```
    Account type: SAVINGS
        Customer name: Bob
        Account balance: $300.00
Account number: 144986167
    Account type: CREDIT_CARD
        Customer name: Jack
        Amount owed: $550.00

Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
time

How months should pass?>
3

Bob earned $1.65 in interest in 3 months. Account Balance is now $301.65


Jack charged $5.50 in interest after 1 months. Account Balance is now $555.50


Credit Card bill sent to customer Jack for account number 144986167 in the amount of $555.

Jack charged $5.55 in interest after 1 months. Account Balance is now $561.05


Credit Card bill sent to customer Jack for account number 144986167 in the amount of $561.

Jack charged $5.61 in interest after 1 months. Account Balance is now $566.66


Credit Card bill sent to customer Jack for account number 144986167 in the amount of $566.
Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
summary

Bank Summary
Account number: 1594686078
```

```
    Account type: SAVINGS
        Customer name: Bob
        Account balance: $301.65
Account number: 144986167
    Account type: CREDIT_CARD
        Customer name: Jack
        Amount owed: $566.66

Enter one of the following commands
    time - pass certain amount of time
    open - open a new account
    close - close an account
    credit - credit an account
    debit - debit an account
    summary - display current bank accounts
    exit - exit program

What do you want to do?>
```

Submission:

```
% ssh glados.cs.rit.edu # or use queeg.cs.rit.edu if glados is down
# password
# go to the directory where your solution is ...
% try hpb-grd lab4-2 'All files required'
# you can see if your submission was successful:
# try -q hpb-grd lab4-2
```

1.3. Homework 4.3 (10 Points)

Objective: To understand casting

Grading:
Correctness: You can lose up to 40% if your solution is not correct
Quality: You can lose up to 80% if your solution is poorly designed
Testing: You can lose up to 50% if your solution is not well tested
Explanation: You can lose up to 100% if your solution if you can not explain your solution during the grading session

Homework Description:

You have to answer and explain the question in the file *Casting.java*.

Explanation:
It might be useful to draw a diagram in order to understand the relationships.

Your Work:

You can add your comments into the source file.

The source:

```
 1      interface B
 2      {
 3          int a = 0;
 4          void methodB();
 5      }
 6
 7      interface BB
 8      {
 9          int a = 1;
10          void methodBB();
11      }
12
13      class A
14      {
15          int a = 2;
16
17          void parentMethod()
18          {
19              System.out.println("parentMethod");
20          }
21      }
22
23      class AA extends A implements B, BB
24      {
25          int a = 3;
26
27          void childMethod()
28          {
29              System.out.println("childMethod");
30          }
31
32          ;
33
34          public void methodB()
35          {
```

```
36              System.out.println("In methodB");
37          }
38
39          public void methodBB()
40          {
41              System.out.println("In methodBB");
42          }
43      }
44
45      public class Casting
46      {
47
48          public static void main(String[] args)
49          {
50              // here we are creating an instance of class A
51              A a = new A();
52
53              // why is this the only method we can call?
54              a.parentMethod();
55
56              // why doesn't this work?
57              //System.out.println(A.a);
58
59              // here we are creating an instance of class AA
60              AA aa = new AA();
61
62              // class AA doesn't define a parentMethod, how can we call one?
63              aa.parentMethod();
64
65              // how could we override this method?
66              aa.childMethod();
67
68              // Which class does this variable refer to?
69              System.out.println(aa.a);
70
71              // Which class does this variable refer to?
72              System.out.println(((A) aa).a);
73
74              // What forces us to define these methods in the AA class?
75              aa.methodB();
76              aa.methodBB();
77
78              // here we are creating an instance of class AA but what is different abou
79              a = new AA();
80
81              // why do we need to cast this?
82              ((AA) a).childMethod();
83
84              // Which class does this variable refer to?
85              System.out.println(a.a);
86
87              // Which class does this variable refer to?
88              System.out.println(((AA)a).a);
89
```

```
90              // call methodB and methodBB using the variable a
91
92
93              // how can we access these variables from the interfaces?
94              System.out.println(B.a);
95              System.out.println(BB.a);
96
97          }
98      }
```

Source Code: Src/24/Casting.java

Requirements:

You have to name the file *Casting.java*.

Example:

An example of a solution execution:

```
You should not run the code in order to be explain it.
You should only run the code to validate your answer.
```

Submission:

```
% ssh glados.cs.rit.edu # or use queeg.cs.rit.edu if glados is down
# password
# go to the directory where your solution is ...
% try hpb-grd lab4-3 'All files required'
# you can see if your submission was successful:
# try -q hpb-grd lab4-3
```