# American University of Beirut

*Department of Computer Science*
*CMPS 212- Summer 2021/22*
*Lab 03*

**Problem 1: Tracing Polymorphism**

1. Assuming the following classes have been defined:

```
public class First {
    public void method2() {

System.out.println("First2");
    }

    public void method3() {
        method2();
    }
}

public class Second extends First {
    public void method2() {

System.out.println("Second2");
    }
}

public class Third extends Second {
    public void method1() {

System.out.println("Third1");
        super.method2();
    }

    public void method2() {

System.out.println("Third2");
    }
}

public class Fourth extends First {
    public void method1() {

System.out.println("Fourth1");
    }

    public void method2() {

System.out.println("Fourth2");
    }
}
```

What is the output produced by each statement below? If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c". If the statement causes an error, write either "compiler error" or "runtime error" as appropriate.

Statement

```
var1.method2();

var2.method2();

var3.method2();

var4.method2();

var5.method2();

var6.method2();

var1.method3();

var2.method3();

var3.method3();

var4.method3();

var5.method3();

var6.method3();

((Second)var4).method1();

((Third)var4).method1();

((Second)var5).method2();

((First)var5).method3();

((Third)var5).method1();

((First)var6).method3();
```

And that the following variables have been defined:

```
First var1 = new Second();
First var2 = new Third();
First var3 = new Fourth();
Second var4 = new Third();
Object var5 = new Fourth();
Object var6 = new Second();
```

```
((Second)var6).method1();

((Second)var6).method3();
```

## Problem 2: Polymorphism and Composition

### A. Defining the Superclass

Design a class named **Account** that contains:
- A private **int** data field named **id** for the account (default **0** ).
- A private **double** data field named **balance** for the account (default **0** )
- A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0** ).
- A private **Calendar** data field named **dateCreated** that stores the date when the account was created. Use Calendar type object.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id, initial balance and annual interest rate. Within the constructor, assign the value of dateCreated using Calendar().
- The accessor and mutator methods for **id** , **balance** , and **annualInterestRate**.
- The accessor method for **dateCreated**.
- A method named **getMonthlyInterestRate()** that returns the monthly interest rate. **monthlyInterestRate** is **annualInterestRate / 12**.
  Note that **annualInterestRate** is a percentage, e., like 4%. You need to divide it by 100
- The method **getMonthlyInterestAmount()** is to return monthly interest amount, not the interest rate. Monthly interest amount is **balance * monthlyInterestRate**.
- A method named **withdraw** that withdraws a specified amount from the account.
- A method named **deposit** that deposits a specified amount to the account.

Write a test program that creates an **Account** object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4%. Use the **withdraw** method to withdraw $2,500, use the **deposit** method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

### B. Creating the Subclasses

Create two subclasses for checking and saving accounts named as **CheckingAccount** and **SavingsAccount**.
- A checking account has an overdraft limit which is double type variable
- A savings account has issued with a credit card automatically. It holds the 16-digit card number an expiry date (Calendar type object). SavingsAccount class must have a method getCreditBalance that returns a credit balance which is three times of the current balance in the account.

### C. Defining an Array/ArrayList of Account type objects in Main method

To understand polymorphism, you need to define an array/array list of Account type objects based on user-provided option. Your main() method must display the following first:
Press (1) for creating a Checking Account Press (2) for creating a Savings Account
You must create at least 4 account type objects in this manner and perform one deposit and one withdraw operation for each account.
Afterwards, print the followings for each account using the concept of Polymorphism. You must not use any toString() method.

| For a Checking Account: | For a Savings Account: |
|---|---|
| This is a Checking Account<br>Account ID:<br>Date Created:<br>Current Balance:<br>Annual Interest Rate:<br>Monthly Interest Amount:<br>Overdraft Limit: | This is a Savings Account<br>Account ID:<br>Date Created:<br>Current Balance:<br>Annual Interest Rate:<br>Monthly Interest Amount:<br>Credit Card Number:<br>Card Expiry Date:<br>Credit Balance: |

### D. Employee, Composition, and Polymorphism.

Using the Employee class from Lab2. Add to that class an Account, then in the main create an array of 3 employees, one with a regular account, one with a checking account, and one with a saving account. Modify the toString() method of the employee to include the account information then loop over the employee array to print all its elements.

As a reminder the employee class is as follows:
Implement a class ***Employee*** with the following members:
- fields: int baseHours(40), double baseSalary(40,000), int baseVacationDays(10), string baseVacationForm(blue)
- Getters & setters for all fields