CIS368 Spr 20 HW3, 11 programming assignments Due on 02/19 (Wed) 11:59 pm
11.8, 11.10, 11.19, 12.2, 12.4, 12.10, 12.22, 12.31 modified, 12.33 modified, 13.4, 13.6, 13.15

11.8 (New Account class) An **Account** class was specified in Programming Exercise 9.7 as follows.

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). Assume that all accounts have the same interest rate.
A private Date data field named dateCreated that stores the date when the account was created.
A no-arg constructor that creates a default account.
A constructor that creates an account with the specified id and initial balance.
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.
A method named getMonthlyInterest() that returns the monthly interest.
A method named withdraw that withdraws a specified amount from the account.
A method named deposit that deposits a specified amount to the account.

Design a new **Account** class as follows.

- ❑ Add a new data field **name** of the **String** type to store the name of the customer.
- ❑ Add a new constructor that constructs an account with the specified name, id, and balance.
- ❑ Add a new data field named **transaction** whose type is ArrayList that stores the transaction for the accounts. Each transaction is an instance of the **Transaction** class, which is defined in Figure 11.6

Transaction	**The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.**
-date: java.util.Date	The date of this transaction.
-type: char	The type of the transaction, such as “W” for withdrawal, “D” for deposit
-amount:double	The amount of the transaction.
-balance: double	The new balance after this transaction.
-description: String	The description of this transaction.
+Transaction(type:char, amount:double, balance:double, description: String)	Construct a Transaction with the specified date, type, balance and transaction.

Figure 11.6 The transaction class describes a transaction for a bank account.

11.10 (Implements **MyStack** using Inheritance extends ArrayList) In listing 11.10 (see below), **MyStack** is implemented using composition. Define a new stack class that extends **ArrayList**. Draw the UML diagram for the classes then implement **MyStack**. Write a test program that prompts the user to enter five strings and displays them in reverse order.
Please provide the snapshots of your testing results.

Listing 11.10 Mystack.java

```
Import java.util.ArrayList;
public class MyStack {
      private ArrayList<Object> list = new ArrayList<>();
      public boolean isEmpty() { return list.isEmpty(); }
      public int getSize() { return list.size() };
      public Object peek() { return list.get(getSize() - 1); }
      public Object pop() {
            Object o = list.get(getSize()-1);
            list.remove(getSize() - 1);
            Return o;
      }
      public void push(Object o){  list.add(o); }
      @Override
      public String toString() { return "stack: " + list.toString(); }
}
```

11.19 (Bin packing using first fit) The bin packing problem is to pack the objects of various weights into containers. Assume each container can hold a maximum of 10 pounds. The program uses an algorithm that places an object into the first in which it would fit. Your program should prompt the user to enter the total number of objects and the weight of each object. The program displays the total number of containers needed to pack the objects and the contents of each container. Here is a sample run of the program:
Please provide the snapshots of your testing results.

```
Enter the number of objects: 6
Enter the weights of the objects: 7 5 2 3 5 8
Container 1 contains objects with weight 7 2
Container 2 contains objects with weight 5 3
Container 3 contains objects with weight 5
Container 4 contains objects with weight 8
```

Does this program produce an optimal solution, that is, finding the minimum number of containers to pack the objects?

12.2 (InputMismatchException) Write a program that prompts the user to read two integers and displays their sum. Your program should prompt the user to read the number again if the input is incorrect.
Please provide the snapshots of your testing results.

12.4 (IllegalArgumentException) Modify the Loan class in Listing 10.2 to throw IllegalArgumentException if the loan amount, interest rate, or number of years is less than or equal to zero.

Please provide the snapshots of your testing results.

Listing 10.2 Loan.java https://liveexample.pearsoncmg.com/html/Loan.html

12.10 (**OutOfMemoryError**) Write a program that causes the JVM to throw an **OutOfMemoryError** and catches and handles this error.
Please provide the snapshots of your testing results.

12.22 (Replace text) Listing 12.16, ReplaceText.java http://www.cs.armstrong.edu/liang/intro10e/html/ReplaceText.html, gives a program that replaces text in a source file and saves the change into a new file. Revise the program to replace a string in a file with a new string for all files in the specified directory using the following command:
`java Exercise12_22 dir oldString newString`
Please provide the snapshots of your testing results.

12.31 modified
The popularity ranking of the baby names in 2015, 2016, 2017, and 2018 is downloaded from https://www.ssa.gov/oact/babynames/ and saved in
https://github.com/YalingZheng/CIS368Spr20/blob/master/HW3/babynamesranking2015.txt
https://github.com/YalingZheng/CIS368Spr20/blob/master/HW3/babynamesranking2016.txt
https://github.com/YalingZheng/CIS368Spr20/blob/master/HW3/babynamesranking2017.txt
https://github.com/YalingZheng/CIS368Spr20/blob/master/HW3/babynamesranking2018.txt
For example, the first two lines in the file baynamesranking2018.txt are as follows:

```
1     Liam  19,837     Emma  18,688
2     Noah  18,267     Olivia     17,921
```

Therefore, the boy's name Liam and girl's name Emma are ranked #1 and the boy's name Noah and girl's name Olivia are ranked #2; 19,837 boys are named Liam and 18,688 girls are named Emma.
Write a program that prompts the user to enter the year, gender, followed by a name, and displays a ranking of the name for the year. Here is a sample run:

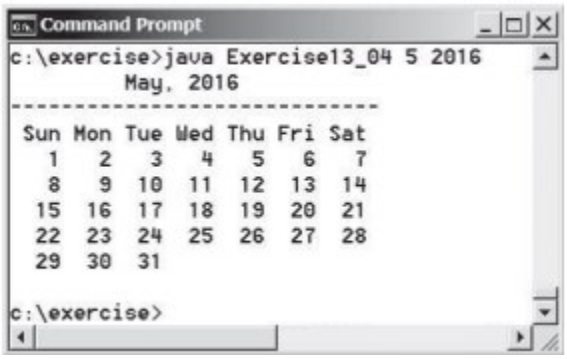
```
Enter the year: 2018
Enter the gender: M
Enter the name: Mason
Mason is ranked #9 in year 2018
```

```
Enter the year: 2015
Enter the gender: F
Enter the name: ABC
ABC is not ranked  in year 2015
```

Please provide the snapshots of your testing results.

12.33 modified.
Modify listing 12.18 WebCrawler.java
http://www.cs.armstrong.edu/liang/intro10e/html/WebCrawler.html to search for the word (e.g. Computer Programming) starting from a URL (e.g., http://cs.armstrong.edu/liang). Your program prompts the user to enter the word and the starting URL and terminates after 10,000 websites have been explored or ten URLs containing the word has been found. Display the first ten URLs that contain the word.
Please provide the snapshots of your testing results.

13.4 (display calendars) Rewrite the PrintCalendar class in Listing 6.12
https://github.com/YalingZheng/CIS368Spr20/blob/master/HW3/PrintCalendar.java to display a calendar for a specified month using the Calendar and GregorainCalendar classes. Your program receives the month and year from the command line. For example
Java Exercise13_04 5 2016
This displays the calendar shown in Figure 13.9.



Please provide the snapshots of your testing results.

13.6 (The **ComparableCircle** class)
Define a class named **ComparableCircle** that extends **Circle** and implements **Comparable**. Draw the UML diagram and implement the **compareTo** method to compare the circles on the basis of area. Write a test class to find the larger of two instances of **ComparableCircle** objects, and the larger between a circle and a rectangle.

Hint: Circle.java, ComparableCircle.java, Rectangle.java, GeometricObject.java

Please provide the snapshots of your testing results.

13.15 (Use BigInteger for the Rational Class)

Redesign and implement the Rational class in Listing 13.13 https://liveexample.pearsoncmg.com/html/Rational.html using BigInteger for the numerator and denominator. Write a test program that prompts the user to enter two rational numbers and displays the results as shown in the following sample run:

```
Enter the first rational number: 3 454
Enter the second rational number: 7 2389
3/454 + 7/2389 = 10345/1084606
3/454 - 7/2389 = 3989/1084606
3/454 * 7/2389 = 21/1084606
3/454 / 7/2389 = 7167/3178
7/2389 is 0.0029300962745918793
```

Please provide the snapshots of your testing results.