RIT | Golisano College of Computing and Information Sciences
School of Information

ISTE-120 - Computational Problem Solving for the Information Domain I
Homework Assignment 2 (HW02)

DELIVERABLE

Zip your files together and submit the zipfile to the myCourses assignment folder for this homework. Homework MUST be submitted on time to receive full credit. Homework submitted to the "late" dropbox will receive a maximum grade of 80%. Homework not submitted to either dropbox will receive no credit.

PROBLEM #1 - Cash Register

Using the CashRegister class provided in myCourses, write code in the main method of the RunCashRegister class to display sales transactions similar to that shown below in Figure 1.

```
 ----jGRASP exec: java RunCashRegister
   Total Sales: $ 70.35
       Payment: $ 80.00
=========================
    Change Due: $ 9.65

   Total Sales: $ 185.10
       Payment: $ 180.00
=========================
Not Enough Payment: $ -5.10


 ----jGRASP: operation complete.
```

Figure 1

DETAILS

1. Download the *CashRegister* folder with the HW2 downloads. Save and unzip these downloads into your HW02 folder.

2. The CashRegister class simulates a very simple cash register whose operations include:
   a. Ring up the sales price for a purchased item.
   b. Enter the amount of payment.
   c. Calculate the amount of change due to the customer.

3. The CashRegister class contains the following set of methods for you to use to record and print sales transactions:
   a. CashRegister()
      - a default constructor method
   b. void recordPurchase(double amount)
      - records the sales amount of an item
   c. void enterPayment(double amount)
      - enters the payment amount received from the customer
   d. void printReceipt()
      - prints a receipt as shown in the above Figure 1, which prints "Total Sales Amount", "Payment", and either "Change Due" or "Not Enough Payment" for a sales transaction.

4. Write Java statements to record the following two transactions into the **main** method of RunCashRegister class:

   Customer #1 (customer1 = new CashRegister();)
   1. item #1 sales price $49.95
   2. item #2 sales price $20.40
   3. Customer paid amount $80

   Customer #2 (customer2 = new CashRegister();)
   1. item #1 sales price $99.95
   2. item #2 sales price $35.15
   3. item #3 sales price $50.00
   4. Customer paid amount $180

5. Once you have successfully written Java code in the **main** method, compile the RunCashRegister class, and execute it. Your terminal window output should be similar to the above Figure 1.

6. Write the following class comments before the RunCashRegister class definition using either line comment (//) or block comment (/* … */).:

   Name: your name
   Course: ISTE-120
   HW: #2-1
   Description: A class to test the CashRegister class

7. Write a brief comment before each customer's transaction statement using either line comment (//) or block comment (/* … */).

8. Create a zip file **(hw02_1.zip)** containing the entire *CashRegister* folder for submission to the dropbox.

PROBLEM #2 – Bank Account

Using the BankAccount class provided in HW02 downloads, write code in the **main** method of the BankTransactions class to display bank transactions similar to that shown below in Figure 2.

```
 ----jGRASP exec: java BankTransactions
Jim's Savings Account Balance is $ 3000.00
=========================================
Jim's Checking Account Balance is $ 0.00
Jim's Checking Account Balance is $ 500.00
Jim's Checking Account Balance is $ 0.00
Jim's Savings Account Balance is $ 2500.00
=========================================
Jen's Checking Account Balance is $ 500.00
Jen's Checking Account Balance is $ 1000.00
Jen's Checking Account Balance is $ 950.00
=========================================
Jim's Savings Account Balance is $ 2000.00

 ----jGRASP: operation complete.
```

Figure 2

DETAILS

1. Locate the *BankAccount* folder in HW02 downloads

2. The BankAccount class simulates a simple bank account whose operations include:
   a. Deposit money
   b. Withdraw money
   c. Inquire the current balance

3. The BankAccount class contains the following set of methods for you to use to record and print bank transactions:
   a. BankAccount (String name)
      - construct a bank account with a zero balance and an account name
   b. BankAccount(String name, double initialBalance)
      - construct a bank account with an initial balance and an account name
   c. void deposit(double amount)
      - deposits money into the bank account
   d. void withdraw(double amount)
      - withdraw money from the bank account

e. double getBalance()
   - gets the current balance of the bank account
f. void transfer(BankAccount ba, double transferAmount)
   - transfer money from another bank account (ba)
g. void printBalance()
   - print the current balance of the bank account
h. void printLineA()
   - print a line of "="s
i. void printLineB()
   - print a line of "&"s

4. Write Java statements to record the following transactions in the same order as written below, into the **main** method of BankTransactions class:

Bank Account Name:	Jim's Savings
Initial Balance:	0
Deposit:	$ 3,000 (note: you only enter 3000 as a method parameter)
Get a Balance of	Jim's Savings
Draw a line of "="s	

Bank Account Name:	Jim's Checking
Initial Balance:	0
Get a Balance of	Jim's Checking
Transfer from Jim's Savings	$ 500
Get a Balance of	Jim's Checking
Withdraw:	$ 500
Get a Balance of	Jim's Checking
Get a Balance of	Jim's Savings
Draw a line of "="s	

Bank Account Name:	Jen's Checking
Initial Balance:	$ 500
Get a Balance of	Jen's Checking
Transfer from Jim's Savings	$ 500
Get a Balance of	Jen's Checking
Withdraw:	$ 50
Get a Balance of	Jen's Checking
Draw a line of "="s	

Get a Balance of	Jim's Savings

5. Once you have successfully written Java code in the run() method, compile the BankTransactions class, instantiate a BankTransactions object, and invoke the run method. Your terminal window output should be similar to the above Figure 2.

6. Write the following class comments before the BankTransactions class definition using either line comment (//) or block comment (/* … */).:

   Name: your name
   Course: ISTE-120
   HW: #2-2
   Description: A class to test the BankAccount class

7. Write a brief comment before each bank account transaction statement using either line comment (//) or block comment (/* … */).

8. Create a zip file **(hw02_2.zip)** containing the entire *BankAccount* folder for submission to the assignment folder.

Name: _____

Homework 2 Grade Sheet

Program 1: RunCashRegister	**Point Value**	**Points Earned**
- Wrote Java statement in **main** method - Instantiated CashRegister objects - Called recordPurchase(double amount) method - Called enterPayment(double amount) method - Called printReceipt() method - Wrote class comments as specified - Wrote comments before each customer transaction statement - Output matches the sample provided	5 10 2 2 2 3 6 10	
-	40	
Program 2: BankTransactions		
- Wrote Java statement in **main** method - Instantiated BankAccount objects with account name - Instantiated BankAccount objects with account name and initial balance - Called deposit(double amount) method - Called withdraw(double amount) method - Called transfer(BankAccount ba, double amount) method - Called printBalance() method - Called printLineA() method - Wrote class comments as specified - Wrote comments before each bank account transaction statement - Output matches the sample provided	5 5 10 3 3 9 3 3 3 6 10	
	60	
Total Points	100	

Note: If your program fails to compile, you will receive a zero for the assignment. A clean compilation means that the compiler generates no warning messages.

Additional Comments: