CIS 162 Project 2
A Simple Cash Register

Due Date

- at the start of lab on June 2, 2015

Before Starting the Project

- Read the following: Chapter 2 Variables and Chapter 5 Branching
- Read section 1.7 (Basic Input) and other related sections in Chapter 2 on reading user input using the Scanner class before completing the challenge requirements
- Read this entire project description before starting

Objectives

After completing this project you should be able to:

- write, compile and run a simple Java class
- write expressions using variables
- write methods to meet specific requirements
- explain the differences between local variables, instance variables and method parameters
- write conditional statements with boolean expressions

Basic Requirements

Create a class, called *CashRegister*, which simulates the functionality of a simple cash register at a local convenience store. You can do simple things like scan prices, make payments, and report daily sales. Simple text messages are displayed on the screen.

- Provide appropriate names and data types for each of the private instance variables: a double for the *current amount due* that changes as the teller scans prices, a double for the *total daily sales* that increases throughout the day, a String for the *store name*, and a **final** double for the *sales tax* (0.06).
- `public CashRegister (String name)` - constructor initializes the instance variables to zero and sets the store name to the provided parameter. Displays a welcome message that includes the store name. (5 pts)
- `public double getTotalSales ( )` - return the total sales for the day. (5 pts)
- `public double getAmountDue ( )` - return the amount due from the current customer. (5 pts)
- `public void scanPrice (double price)` - add the given price to the current amount due. Ignore, for now, if the price is negative. Notice that no Scanner should be used inside this method, the price is entered as parameter. Display the price. (5 pts)
- `public void completeSale ( )` – After the cashier has scanned the last item, calculate the sales tax and add to the current amount due. Add the amount of this sale to the total sales for the day. Display the sales tax and total amount due. (10 pts)
- `public void cancelSale ( )` – set the current amount due to zero and display an appropriate message. (5 pts)

- `public void makePayment (double amount)` – reduce the amount due by the amount paid and display the amount of change due to the customer. Reset amount due to zero. Ignore, for now, if the customer does not pay enough. (10 pts)
- `public void showSalesReport ( )` – display the store name and total sales for the day (5 pts)
- `public static void main ( )` – as defined below in the software testing section (10 pts)
- Create a new instance variable of type `NumberFormat` (will be explained during lecture) and instantiate it in the constructor. Use this object throughout all methods to properly display currency amount (10 pts)

`NumberFormat`	
Include this package at the beginning of your class	`import java.text.NumberFormat;`
To declare fmt as a NumberFormat object	`private static NumberFormat fmt;`
To instantiate fmt	`fmt = NumberFormat.getCurrencyInstance();`
Example of printing amount as currency	`double amount = 5.6;` `System.out.println(fmt.format(amount));`

Challenge Requirements

The following should only be attempted after all basic requirements have been completed.

- Modify the `scanPrice (double price)` method to update the amount due only if the price, entered as parameter, is above zero. Otherwise, print an error message stating there was a scanning error. (5 pts)
- `public void clearAllSales ( )` – Prompt the cashier if she really wants to clear all sales. Read the response from the keyboard. If she types "y", set the total daily sales to zero. Otherwise, make no changes. Display an appropriate message. (5 pts)
- Modify the `makePayment ()` to allow for several messages. 1) If the payment is negative, 2) if the payment is not enough, 3) if the payment is too much and 4) if the payment is exact. Each message should be unique and only one is displayed. (10 pts)

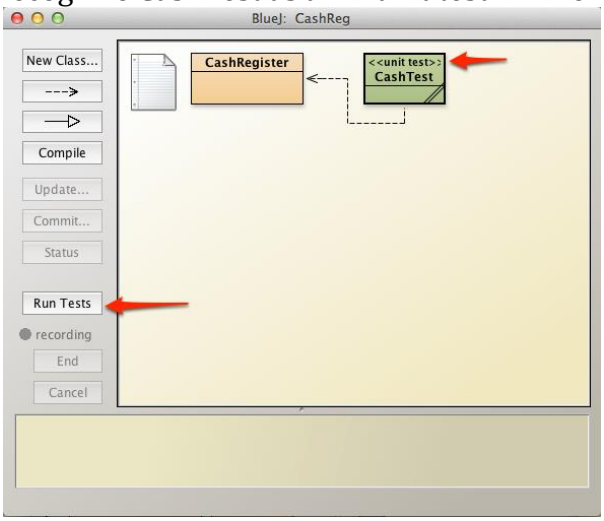
Software Testing

Software developers must plan from the beginning that their solution is correct. BlueJ allows you to instantiate objects and invoke individual methods. You can carefully check each method and compare actual results with expected results. However, this gets tedious and cannot be automated.

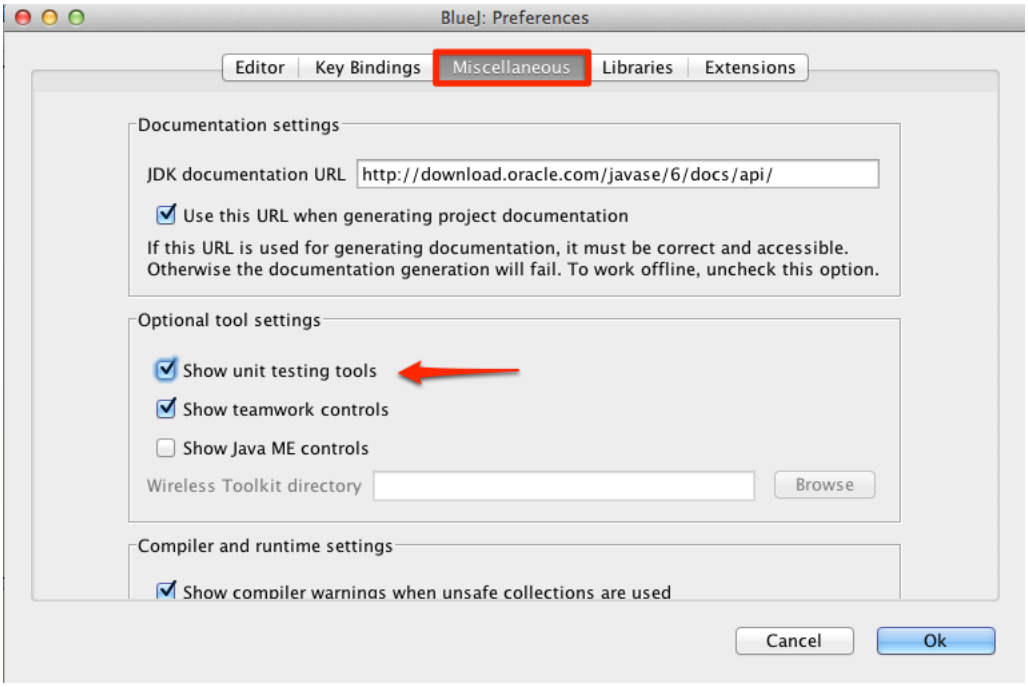
JUnit Testing

JUnit is a Java library that provides a framework to automate software testing. A JUnit test file `CashTest.java` has been provided by your instructor. Use the following instructions to run JUnit Testing

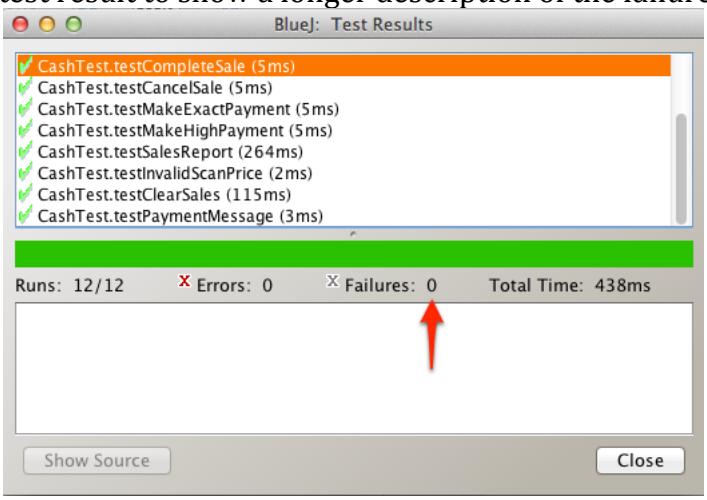
1. Name your class CashRegister. **Exact spelling is required**. Using a different name (cashRegister, Cashregister, CashReg, …) will fail the test
2. Complete both the basic and challenge requirements describe above
3. Download CashTest.java to the same folder as your CashRegister.java
4. Restart Bluej (if CashTest does not show up on your BlueJ window). BlueJ should recognize CashTest as a "<<unit test>>" file. Otherwise, continue to Step 7.

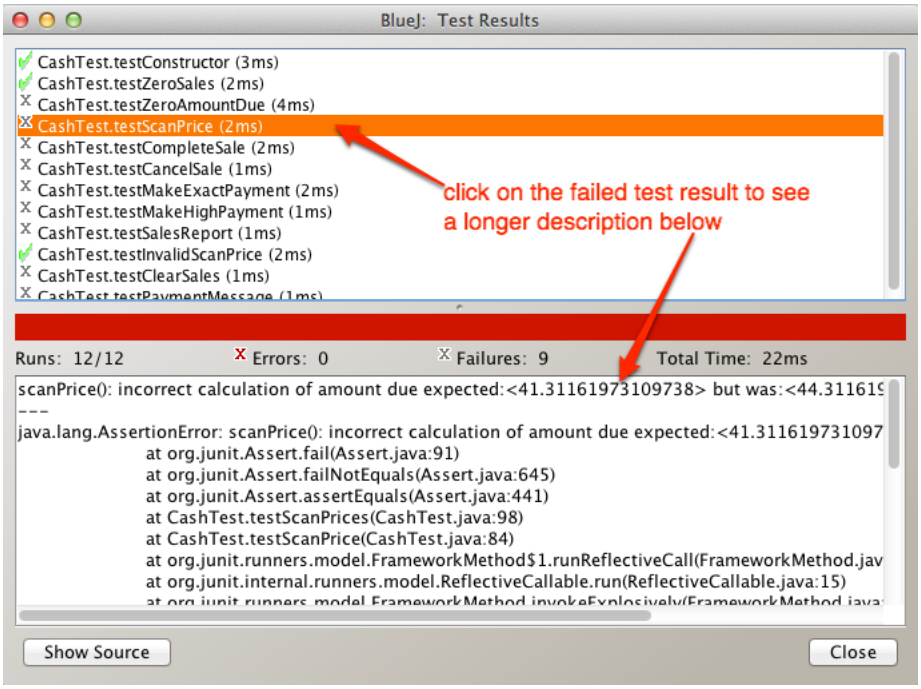


5. If the "Run Tests" button is not visible, go to Bluej Preferences. Under the "Miscellaneous" tab check the "Show unit testing tools"



6. On the BlueJ window, click the "Run Tests" button. If your class passes all the test cases, the test results window will show a green horizontal bar and zero failures. Otherwise, you will see a red horizontal bar and the number of failures. Click on a test result to show a longer description of the failure.





7. Run the main method() of CashTest.java. The test results will be printed on BlueJ terminal window.

Testing From Your Own main() method

Another approach is to write a `main` method that calls all the other methods in a carefully designed sequence.

For this project, write a main method that instantiates at least two cash registers for different stores and invokes each of the methods with a variety of parameter values to test each method. It takes careful consideration to anticipate and test every possibility.

A brief and incomplete example is provided below. Your method will be longer and instantiate two cash registers for different stores.

```
public static void main(String [] args)
{
    CashRegister uMart = new CashRegister("uMart");
    uMart.scanPrice(1.50);
    uMart.scanPrice(3.00);
    uMart.completeSale();
    uMart.makePayment(5.00);
}
```

The sample main method will create the following output to the terminal window:

```
Welcome to uMart!
Price: $1.50
Price: $3.00
Sales Tax: $0.27
Amount Due: $4.77
Payment: $5.00
Thank You! Your change is $0.23
```

Testing Using Instructor Provided Test Cases

Sample Messages

The following messages are provided as examples. Your messages can be more creative as long as they convey the correct information.

Greeting message when object instantiated	`Welcome to Scott's Corner Store!`
After cashier scans an item	`Price: $24.99`
If cashier scans a negative amount	`Scanning error. Please try again.`
After cashier completes the sale	`Sales Tax: $1.20` `Amount Due: $24.99`
If customer pays a negative amount	`We do not accept credit cards`
If the current sale is cancelled	`Sale cancelled`
If customer pays exact amount	`Payment: $5.00` `Thank You. Have a nice day!`
If customer pays too much	`Payment: $5.00` `Thank You. Your change is $1.45`
If customer pays too little	`Payment: $5.00` `You still owe $1.45`
If all sales are cleared	`Are you sure you want to cancel all sales? (y/n)` `All sales cancelled`
Daily Sales Report	`Scott's Corner Store` `Total Daily Sales: $1,577.15`

Grading Criteria

There is a 50% penalty on programming projects if your solution does not compile.

- Stapled cover page with your name and signed pledge. (-5 pts if missing)
- Project requirements as specified above. (90 pts)
- Elegant source code that follows the GVSU Java Style Guide. (10 pts)

Late Policy

Projects are due at the START of the class period. However, you are encouraged to complete a project even if you must turn it in late.

- The first 24 hours (-20 pts)
- Each subsequent weekday is an additional -10 pts
- Weekends and university holidays are free days.

Turn In

A professional document **is stapled** with an attractive cover page. Do not expect the lab to have a working stapler!

- Cover page - Provide a cover page that includes your name, a title, and an appropriate picture or clip art for the project
- Signed Pledge – The cover page must include the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor)." You are responsible for understanding and adhering to the School of CIS Guidelines for Academic Honesty.
- Time Card – The cover page must also include a brief statement of how much time you spent on the project. For example, "I spent 7 hours on this project from September 22-27 reading the book, designing a solution, writing code, fixing errors and putting together the printed document."
- Sample Output – a printout of the Terminal window after running the main method that shows a variety of the printed messages. You can cut and paste into the Word document that contains your cover page.
- Source code - a printout of your elegant source code printed from BlueJ with line numbers (with your name).