CSCU9A2 Practical 10A (week 12) Spring 2017
Object Oriented Programming 2 3rd/4th April

If you get stuck or need help at any time, ask a demonstrator.

Remember: All checkpoints must be completed by the end of this week.

THIS WORKSHEET:

This worksheet develops an object oriented version of the Cash Register application from Practical 4A.

THE CASH REGISTER AS A SEPARATE CLASS FROM THE GUI

- Copy the *folder* **CashRegisterOO** from **V:\CSCU9A2\Java** to your **CSCU9A2** folder on **H:**.
- Launch BlueJ and create a new BlueJ project in your **CashRegisterOO** folder.
- Open the **CashRegister** class in the BlueJ editor and read it carefully. This is closely based on the **CashRegister** example studied in lectures – each instance of the class is designed to hold the information about one supermarket cash register: the current total price so far for the current customer, and the number of items purchased. The two private instance (global) variables `totalPrice` and `itemCount` hold the key data, and there are three public methods to be called by a main program to manage the data.
- Compile the **CashRegister** class. Note: the **CashRegisterGUI** will **not** compile correctly yet, but that will not prevent you exercising **CashRegister**.
- Take a few minutes to exercise the **CashRegister** class using *BlueJ's object workbench* (see practical worksheet 9B): Instantiate a couple of cash register objects, add some items, check the totals, and look at the objects using the object inspector.
- Open **CashRegisterGUI** in the BlueJ editor. This is the main program, read it carefully: It creates a user interface for adding items to the cash register and displaying the total price and item count so far. It is very closely based on the **CashRegister** application from practical 4A. However, the parts of the previous application responsible for the cash register data (some variables and methods) have been moved to the new **CashRegister** class, and the main program instead declares a **CashRegister** variable on line 40, then places a new instance of **CashRegister** in that variable on line 102 (in `setUpData`, called from `main`).
- While you are reading **CashRegisterGUI**, notice that, in `createGUI`, a `GridLayout` is configured for the window instead of `FlowLayout`. This will arrange the GUI widgets in a two column, five row layout: in the order in which they are added to the window, they fill the top row from left to right, then the second row, etc. In two places, dummy (empty) **JLabel**s have been added so that some places in the grid are not used. Widgets are automatically resized to fill their place in the grid. `GridLayout` gives some more control over some aspects of window layout, but it is quite simple and the effect is sometimes a bit ugly due to the resizing. *(Sometime: Try a Google search for "Java layouts" to find information about other layout managers.)*
- Now try compiling **CashRegisterGUI**. There is an error on line 115: BlueJ reports *"cannot find symbol – method addItem(double)"*. Although there *is* a method called `addItem`, the compiler is correct as it is looking for it in the **CashRegisterGUI** class, whereas it is actually in the **CashRegister** class. You should correct the problem by inserting `cashRegister.` in front of the method call in line 115 – `cashRegister` is the name of a variable containing a **CashRegister** object, and the compiler (& JVM) will now *look in that object* for the `addItem` method to call.

- Compile **CashRegisterGUI** again. There are a few more similar errors to correct. One of the errors, however, is more serious as there is a required method *missing* from the **CashRegister** class, and adding `cashRegister.` in front of the method call does not completely solve the problem. You should add a suitable new method to the **CashRegister** class (it is a very simple method).
- Now the program should compile and run correctly. You can add a few items, view the results, and see the effect of the `GridLayout`.
- Now add one more button ("New customer") into the *right column in the fifth row* of the window layout: This is to be clicked when starting to process a new customer's purchases, and should set the total price and item count in the cash register (and in the display) to 0. You will need to declare the button, create it, add it into the correct place in the GUI, and handle its click in `actionPerformed` (so, some `if` statements are needed). You will also need to add one or two methods to **CashRegister** for 0ing the totals, to be called from `actionPerformed`.

Remember that your code should be neatly formatted and commented.

- **Checkpoint [OOP 2]: Show a demonstrator your cash register working correctly, including the New customer button. Answer any questions they ask you.**

ADDING ANOTHER CASH REGISTER

Now you will add a second cash register to the main application. Using the Power of Object Orientation, you will *not* need to duplicate the cash register key data variables and methods – **you will only need to make changes to the GUI**. [Perhaps the application should now be called **SupermarketGUI** rather than **CashRegisterGUI**, but you do not need to change it!]

- Declare one more **CashRegister** instance variable to hold the details of a second cash register.
- In `setUpdata`, assign the new variable *a new instance* of **CashRegister**.
- Display its total price and item count in the right column, alongside the first cash register's totals.
- Add a new button "Add item to register 2" in the right column alongside the "Add item button". When this button is clicked, the item being entered should be added to the second cash register (and the display should be updated).
- Finally, arrange for *two* "New customer" buttons, clearing their respective cash registers.

Remember that your code should be neatly formatted and commented.

SBJ March 2017