CPSC 101 Winter 2016
Midterm II—11 March 2016

Name (Printed) : ______________________

Signature : ______________________

Student Number : 2 3 0 _ _ _ _ _ _

Question	Score
1	/10
2	/3
3	/2
4	/5
5	/2
6	/1
7	/1
8	/1
9	/3
10	/4
11	/4
12	/2
13	/5
14	/2
15	/5
Total	/50

- This is a **50** minute exam. This exam contains **8** pages of questions not including this cover page. Make sure that you have all of them.
- Put your name on the top right hand corner of each page as examination papers sometimes come unstapled.
- Non-programmable calculators and simple wrist-watches are allowed. **Cellphones and other non-medical electronic devices are prohibed**.
- Answer all questions on the exam sheet. If you do some of your work on the back of a page, clearly indicate to the marker what work corresponds with which question.
- Partial marks shall be awarded for clearly identified work.
- *Read each question carefully. Ask yourself what the point of the question is. Check to make sure that you have answered the question asked.*
- This exam counts as **15%** of your total grade. There are **50** points total on the exam.

CPSC 101 *Winter 2016* Name:

True False

1 each **1.** Circle **TRUE** or **FALSE** as appropriate. Questions that don't clearly indicate *one* choice shall be marked wrong. If you feel that the answer depends on how you interpret the question, give a brief reason for the answer you chose.

(a) JAVA objects never contain other objects. **TRUE** **FALSE**

(b) If a class contains a final method, the class must be final. **TRUE** **FALSE**

(c) The JAVA garbage-collector removes unused objects from the stack. **TRUE** **FALSE**

(d) A JAVA graphics program doesn't necessarily end when its `public static void main(String [] args)` exits. **TRUE** **FALSE**

(e) The `java.awt.*` classes are newer than the `javax.swing.*` classes. **TRUE** **FALSE**

(f) The JAVA Swing libraries are designed to operate on a single thread. **TRUE** **FALSE**

(g) In JAVA, a class can be declared inside a method. **TRUE** **FALSE**

(h) In JAVA, methods of an inner (non-static nested) class can access private member variables of an object of the containing class. **TRUE** **FALSE**

(i) JAVA explicitly supports concurrency. **TRUE** **FALSE**

(j) The `equals` method of the `Object` class is `final`. **TRUE** **FALSE**

Memory Organization

(1) **2.** (a) What is the name of the region of memory that stores variables and arguments to functions?

(1) (b) What is the name of the region of memory that stores objects?

(1) (c) What is a third region of memory used by a running JAVA program.

3. Speaking of JAVA,

(1) (a) what causes an object to be created in memory?

(1) (b) What causes an objects to be removed from memory?

(1) **4.** (a) When are stack frames created?

(1) (b) When are stack frames destroyed?

(2) (c) Name two kinds of information stored in a stack frame.

(1) (d) What is the difference between the stack frame corresponding to a `static` method and the stack frame corresponding to a non-`static` method?

Graphics and User Interfaces

(2) **5.** Consider the code fragment

```
1   private static void createAndShowGUI() {
2      JFrame frame = new JFrame("RadioButtonDemo");
3      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
4      ...
```

Explain what line 3 does and why it is important to put it in simple GUI applications.

For the questions below, choose the *best* answer possible, and clearly indicate *one* choice. Supply reasons to the right if you are not sure, or think that the question is open to multiple interpretations.

(1) **6.** The `JFrame` class is

(**a**) a framework class for building a GUI.
(**b**) an interface.
(**c**) a top-level window class provided by the `javax.swing` libraries.
(**d**) a top-level window class provided by the `java.awt` libraries.

(1) **7.** The `javax.swing.SwingUtilities.invokeLater` is

(**a**) a static method for ensuring that your application quits when its main window does.
(**b**) a non-static method for ensuring that your application quits when its main window does.
(**c**) a static method that invokes its argument on the Swing GUI thread.
(**d**) a non-static method that invokes its argument on the Swing GUI thread.

(1) **8.** An *adapter*, such as `MouseAdapter`, is

(**a**) something that converts mouse clicks into button presses.
(**b**) an abstract class related to a listener.
(**c**) a class that implements the corresponding listener with do-nothing methods.
(**d**) an interface that provides more functionality than most listeners do.

---

(3) **9.** Explain what the *model-view-controller* idea is about.

- What is one possible advantage of separating the model and the view?
- Does the Java Swing library use this idea?

(4) **10.** In Figure 1 on page 8, there are references to the imported classes `JPanel`, `Color`, and `Graphics`? Which of these classes are likely imported from `javax.swing`, and which are likely imported from `java.awt`? What else should JAVA-programmers using graphics know about `java.awt.*` and `javax.swing.*`?

(4) **11.** In order that clicking the mouse on the `CounterPanel` causes the count to increment, what code (be as precise as possible) needs to be added *at* Line 15 in Figure 1 on page 8 ?

If you need other code elsewhere to make this work indicate what that code is.

(1) **12.** (a) In Figure 1 on page 8, what kinds of things can cause the `paintComponent` method to be called?

(1) (b) Suppose that users want to cause part of a GUI to be redrawn. Instead of calling the `paintComponent` method directly, what should they do?

13. Suppose that you have a `CarCollectionPanel` that subclasses from `JPanel`, and that the `CarCollectionPanel` adds child objects that are `CarComponent`s.

(1) (a) The method to add a child graphics object is actually found in `java.awt.Container`. What does this say about the relationshiop between `JPanel` and `java.awt.Container`?

(1) (b) The argument of the `add` method is declared to be a `java.awt.Component`. What does this say about the parent classes of `CarComponent`?

(1) (c) What class are you likely to extend in creating a `CarComponent`?

(2) (d) The documentation for the `add` method says that it notifies the layout manager when you add a child. Explain what a layout manager is, and how you tell `JPanel` what choice of layout manager you want.

Other Questions

(2) **14.** What are two methods of the `Object` class that are commonly overridden?

(3) **15.** (a) In practising Object Oriented Design, ABC's refer to Attributes, Behaviours, and Collaborations. Explain what these are.

(2) (b) Compare and contrast *private member variables* and *attributes*.

Figure 1: Sample code for Questions 12

```
CounterPanel.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/* comments that start with "///" indicate missing code */

public class CounterPanel extends JPanel {
  private int myCounter ;
  public void increment() { ++myCounter ; }
  public int  getCounter() { return myCounter ; }

  public CounterPanel () {
    setBackground(Color.black) ;
    setForeground(new Color(255,63,63)) ;
    /// set things up so that clicking the mouse increments the count
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g) ;
    setFont(new Font("SansSerif",Font.PLAIN,35)) ;
    g.drawString(""+getCounter(),getWidth()/2-10,getHeight()/2+5) ;
  }

  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(
        /// code that calls createAndStartGui() ;
        );
  }

  private static void createAndStartGui() {
    /// set up a JFrame called aFrame
    aFrame.add(new CounterPanel()) ;
    aFrame.setVisible(true) ;
  }
}
```