CSCI 136 Written Exam #2 **Name: _____**
Fundamentals of Computer Science II
Spring 2013

This exam consists of 5 problems on the following 7 pages.

You may use your double-sided hand-written 8 ½ x 11 note sheet during the exam. No computers, mobile devices, cell phones, or other communication devices of any kind are permitted.

If you have a question, raise your hand and I will stop by. Since partial credit is possible, **please write legibly and show your work**.

Problem	Points	Score
1	8	
2	13	
3	16	
4	20	
5	9	
Total	66	

1. **Classes, loops** (8 points). Consider the following program:

```
public class Prob1
{
    private static int n = 0;

    public Prob1(int a)       { n += a;        }
    public String toString()  { return "" + n; }

    public static void main(String[] args)
    {
        Prob1 p = new Prob1(1);
        for (String s : args)
            p = new Prob1(Integer.parseInt(s));
        System.out.println(p);
    }
}
```

Below are four example executions of the program. Give the output produced by the program. If the given input would cause a runtime error, write "runtime error".

Command line	Output
`% java Prob1`	1
`% java Prob1 1`	2
`% java Prob1 1 10.0`	runtime error
`% java Prob1 -1 123`	123

2. **Web, regular expression** (13 points). The US census bureau has a web page with a text listing the population of various cities in different US states. The page also contains headings and other text you don't care about. Here is a representative sample:

```
=== Population as of 5/1/13 ===
New York:
New York, NY: 8,244,910
Pleasantville, NY: 7,055

Minnesota:
Minneapolis, MN: 387,753
Saint Paul, MN: 288,448

Montana:
Butte, MT: 33,704
Anaconda, MT: 9,299
Eureka, MT: 998
```

You are building a program that takes as input a two letter uppercase state code and outputs the total state population according to the page. Assume cities only consist of the letters A-Z (upper or lowercase) and 0 or more spaces. The census bureau always uppercases state abbreviations. Place letters in the underlined spaces to create a working implementation. ***Not all letters will be used.***

```
public static void main(String [] args)
{
   String state = args[0];
   long sum = 0;
   _C_
   {
      URL url  = new URL("http://census.gov/bystate.txt");

      URLConnection     c = _I_;
      InputStreamReader s = new InputStreamReader(_G_);
      BufferedReader    r = new BufferedReader(s);

      while (true)
      {
         String line = r.readLine();
         if (_Y_) break;
         String regex = _N_ + state + _R_;
         if (line._U_(regex))
         {
            String [] cols = line.split(":");
            cols[1] = cols[1].replaceAll(_M_, "");
            sum += Integer.parseInt(cols[1]);
         }
      }
   }
   _E_
   {
      e.printStackTrace();
   }
   System.out.printf(_J_, state, sum);
}
```

A. `while (sum == 0)`
B. `while (true)`
C. `try`
D. `catch`
E. `catch (Exception e)`
F. `catch ()`
G. `c.getInputStream()`
H. `new Socket(url, 80);`
I. `url.openConnection()`
J. `"Population %s: %d\n"`
K. `"Population %d: %s\n"`
L. `"(\\s,)"`
M. `"[\\s,]"`
N. `"[a-zA-Z ]+, "`
O. `"[A-Z a-z], "`
P. `"[a-z, ]"`
Q. `": [0-9]+"`
R. `": [0-9,]+"`
S. `"[123456789]+"`
T. `toUpperCase`
U. `matches`
V. `equals`
W. `equalsIgnoreCase`
X. `line.eof`
Y. `line == null`
Z. `line != state`

3. **Multiple choice** (2 points each). For each question, circle the ***ONE*** best answer.

a) You want to create a program to compute all primes from 2 to 2 billion. Which type of programming language is likely to be significantly slower than the others on a given computer?

I. Assembly language
II. Interpreted language
III. Language compiled to byte code and executed by a JIT compiler
IV. Language compiled to native code
V. They all perform about the same

b) Assume `foo()` is a `synchronized` non-static method in the class `Widget` and `w` is an object of type `Widget`. Which of the following is ***TRUE***:

I. The synchronized keyword reduces the amount of time required by the method.
II. The synchronized keyword reduces the amount of memory required by the method.
III. Before `w.foo()` can start, no thread can be executing a synchronized method in object `w`.
IV. Before `w.foo()` can start, no thread can be executing a synchronized method in any object of type `Widget`.
V. Before `w.foo()` can start, Java ensures that the method cannot cause deadlock.

c) Which of the following statements about Java classes is ***TRUE***:

I. A class that **`implements`** obtains all its parent's implemented methods and instance variables.
II. A class can **`extends`** one or more abstract base classes.
III. A class can **`extends`** one or more concrete base classes.
IV. A class that **`implements`** must implement at least one of the methods in the specified interface.
V. A class that **`implements`** must implement all the methods in the specified interface.

d) Which of the following statements most accurately describes what the `Activity` and `Intent` data types represent in Android mobile development?

I. An `Intent` makes use of one or more `Activity` objects to achieve the application's overall goal.
II. An `Activity` typically represents a single screen in the user interface. An `Intent` can be used to pass a message between different `Activity` objects.
III. An `Activity` might be used to send a message causing the phone dialer to open up with a specific number.
IV. The user interface of an Android application is created by extending the `JFrame` class.
V. The user interface of an Android application is created by extending the `JPanel` class.

e) Which of the following statements about Java client and server socket programs is ***FALSE***:

I. A client specifies the destination host via either a numeric IP address or by a domain name.
II. A client specifies the target application (e.g. web, email) on the destination host via a numeric port number.
III. A server must specify a numeric port number that it is listening on but is not required to specify an IP address or domain name.
IV. Before data can be exchanged, a new connection must undergo a three-way handshake (at least for the TCP connection-oriented sockets we learned about in class).
V. The client program must be multi-threaded so multiple users can use the server program at the same time.

f) Consider the following method that reads a single integer from the specified file:

```
public static int readInt(String filename)
{
   try
   {
      Scanner scanner = new Scanner(new File(filename));
      return scanner.nextInt();
   }
   catch (FileNotFoundException e)
   {
      return -1;
   }
}
```

The above method compiles and works as described. Now assume you have removed the **try-catch** block. Which of the following would make the method compile again?

I. Nothing needs to be done, `FileNotFoundException` is an unchecked exception and catching it is optional.
II. Before creating the `Scanner`, add an **if** statement to check if the file exists. If it does not, **return** `-1`
III. Compile it on the command line instead of from Eclipse.
IV. There is no way to make this compile without a **try-catch** block.
V. Add **throws** `FileNotFoundException` to the method declaration (after the parameter list).

g) Consider the following code:

```
Duck [] ducks;
Duck [] ducks2 = new Duck[3];

Duck meanDuck = null;
Duck loneDuck = new Duck();
meanDuck = loneDuck;

ducks2[(int) (Math.random() * 3)] = new Duck();

System.out.println("Quack quack!");
```

How many `Duck` object instances are on the heap on the final `System.out.println` line?

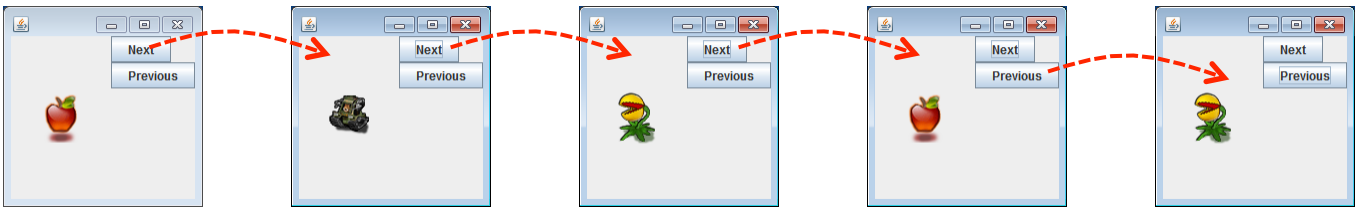
I. 0
II. 1
III. 2
IV. 3
V. 5

h) Java has the built-in classes such as `Double`, `Integer`, `Long`, and `Boolean`. Which of the following is a ***VALID*** reason to use a wrapper class instead of its primitive counterpart?

I. They reduce the memory footprint of your program.
II. They increase the speed of your program.
III. They allow values to be stored in Java generic classes such as those in the collections API.
IV. They provide better type safety via auto-boxing and auto-unboxing.
V. They protect the programmer from common mistakes such as division by zero.

4. **GUIs** (16 points). You are building a Java desktop GUI application that allows the user to specify one or more images as command line arguments. The GUI then displays the first image and the user can click a "Next" or "Previous" button to cycle through the images. Here is an example run where red dashed arrows show the button that was clicked:

```
% java ImageBrowser Apple.png Rover.png FlyTrap.png
```



a) Place letters into the underlined spaces in order to create the GUI shown above. For now, ignore the details of what is inside the inner-classes `ImagePanel`, `NextListener`, and `PrevListener`. ***Not all letters will be used and some may be used more than once.***

```
public class ImageBrowser _U_
{
   private Image [] images;
   private int current = 0;
   private class ImagePanel     _V_   { /* Display image    */ }
   private class NextListener   _X_   { /* Next button hit */ }
   private class PrevListener   _X_   { /* Prev button hit */ }

   public ImageBrowser(String [] args)
   {
      images = _R_;
      for (int i = 0; i < args.length; i++)
         images[i] = (_T_).getImage();

      getContentPane().add(new ImagePanel(), _D_);

      JButton buttonNext = new JButton("Next");
      JButton buttonPrev = new JButton("Previous");

      buttonNext.addActionListener(_N_);

      buttonPrev.addActionListener(_O_);

      _L_ panel = new _L_();

      panel.setLayout(new _H_(panel, _H_.Y_AXIS));
      panel.add(buttonNext);
      panel.add(buttonPrev);

      _E_.add(panel, _C_);

      setSize(200, 200);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);
   }

   public static void main(String [] args)
   { ImageBrowser browser = new ImageBrowser(args); }
}
```

A. BorderLayout.*NORTH*
B. BorderLayout.*SOUTH*
C. BorderLayout.*EAST*
D. BorderLayout.*CENTER*
E. getContentPane()
F. addMouseListener
G. add
H. BoxLayout
I. BorderLayout
J. FlowLayout
K. JApplet
L. JPanel
M. JFrame
N. new NextListener()
O. new PrevListener()
P. new ImagePanel()
Q. new Image()
R. new Image[args.length]
S. new Image[args.length-1]
T. new ImageIcon(args[i])
U. extends JFrame
V. extends JPanel
W. extends JApplet
X. implements ActionListener
Y. implements JFrame
Z. implements MouseListener

4. **GUIs** (continued)
b) Now that you have the GUI from the previous problem all laid out, complete the code to make the button handlers and image display work as shown in the diagram. The image should be centered vertically and horizontally in the panel. You can assume images are smaller than the size of the panel. As a reminder, here are some methods you may need from various GUI and graphics related classes:

```
  class Graphics   boolean drawImage(Image img, int x, int y, ImageObserver observer)
  class Image      int getWidth(ImageObserver observer)
  class Image      int getHeight(ImageObserver observer)
  class JPanel     int getWidth()
  class JPanel     int getHeight()
```

```
private class ImagePanel /* answer from part a */
{
   public void paintComponent(Graphics g)
   {
      g.drawImage(images[current],
                  getWidth()  / 2 - images[current].getWidth(this)  / 2,
                  getHeight() / 2 - images[current].getHeight(this) / 2,
                  this);
   }
}

private class NextListener /* answer from part a */
{
   public void actionPerformed(ActionEvent event)
   {
      current++;
      if (current >= images.length)
         current = 0;

      repaint();
   }
}

private class PrevListener /* answer from part a */
{
   public void actionPerformed(ActionEvent event)
   {
      current--;
      if (current < 0)
         current = images.length - 1;

      repaint();
   }
}
```

5. **Style** (9 points). List three coding style principles or habits that help prevent bugs and makes your code more maintainable/understandable. Briefly describe why this style/habit is a good idea.

Many possible answers, here are a selection of some typical answers:

Style/habit	**Why is this a good idea?**
Comments	Lets others understand in plain English what part of your program does. Commenting before writing tricky bits of code often helps you formulate a game plan.
Descriptive variable names	Make code easier to read, leads to fewer bugs resulting from using one variable when you intended another.
Avoiding repeated code	Repeating code means if you find a bug, you have to remember to change it in every single location. Also repeated code makes the program longer and harder to understand- better to consolidate repeated code in a single helper method.
Avoiding magic numbers	While a constant value may have meant something to you when you wrote the code, others may not understand what the magic number actually means. Repeating a magic number means if the number needs to be changed you have to change it everywhere. Better to use a static final constant instance variable instead.
Indenting code consistently	Makes code easier to read and helps you understand the structure of the conditionals and loops.
Using ample whitespace	Whitespace can be used to separate logical section of a program or to make a line of code more understandable.
Sticking to a single bracing style	Switching between different bracing styles makes the code structure harder to see.