ECE 30862 Fall 2014, Second Exam

DO NOT START WORKING ON THIS UNTIL TOLD TO DO SO. LEAVE IT ON THE DESK.

THE LAST PAGE IS THE ANSWER SHEET. TEAR IT OFF AND PUT ALL ANSWERS THERE. TURN IN BOTH PARTS OF THE TEST WHEN FINISHED.

You have until 9:00PM to take this exam. The first 5 questions are worth 5 points and the rest are worth 2.5 points. The total number of points should be 100. After taking the test turn in both the test and the answer sheet.

Your exam should have 9 (nine) pages total (including this cover page and the answer sheet). As soon as the test begins, check that your exam is complete and *let the proctors know immediately if it is not.*

This exam is open book, open notes, but absolutely no electronics. If you have a question, please ask for clarification. If the question is not resolved, state on the test whatever assumptions you need to make to answer the question, and answer it under those assumptions. *Check the front board occasionally for corrections.*

I have neither given nor received help during this exam from any other person or electronic source, and I understand that if I have I will be guilty of cheating and will fail the exam and perhaps the course.

Name (must be signed to be graded):

Name

Last four digits of your ID:

Multithreading questions.

```
class MyThread extends Thread {

   public static int count = 0;
   public static Object lock = new Object( );

   public MyThread( ) { };

   public void run( ) {
      synchronized(lock) {
         count++;
      }
   }
}
```

```
class Main {

   public static void main(String args[]) {

      MyThread[ ] threads = new MyThread[4];
      for (int i = 0; i < 3; i++) {
         threads[i] = new MyThread( );
      }

      for (int i = 0; i < 3; i++) {
         threads[i].start( );
         threads[i].run( );
      }

      for (int i = 0; i < 3; i++) {
         try {
            threads[i].join( );
         } catch ( java.lang.InterruptedException e)
           {System.out.println(e);}
      }

      System.out.println(MyThread.count);
   }
}
```

Q1 What is the value printed by the statement *System.out.println(MyThread.count);* in *Main*?

Q2 If the line *synchronized(lock)* {, and the closing }" is removed from the *run* method of the *MyThread* class, what will be printed by the statement *System.out.println(MyThread.count);* in *Main*?

a. There is a race and so in Java any value could have been printed.

b. There is a race and multiple threads can read the same value before incrementing, causing updates to be lost. The value printed can be less than the value with the **synchronize** statement.

c. There is a race and a thread with a lower value of **count** can overwrite a higher value of count written previously, but the value printed can be less than the value with the **synchronize** statement.

d. The **synchronize** is unecessary and the value printed will be the same as in Q1.

e. b and c are both correct.

Q3: Would the program give the same answer if the line *synchronized(lock)* {, and the closing }", is removed from the *run* method of the *MyThread* class and the *run* method is made a *synchronized* method?

This is same program as was on the previous page.

```
class MyThread extends Thread {

   public static int count = 0;
   public static Object lock = new Object( );

   public MyThread( ) { };

   public void run( ) {
      synchronized(lock) {
         count++;
      }
   }
}
```

```
class Main {

   public static void main(String args[]) {

      MyThread[ ] threads = new MyThread[4];
      for (int i = 0; i < 3; i++) {
         threads[i] = new MyThread( );
      }

      for (int i = 0; i < 3; i++) {
         threads[i].start( );
         threads[i].run( );
      }

      for (int i = 0; i < 3; i++) {
         try {
            threads[i].join( );
         } catch ( java.lang.InterruptedException e)
           {System.out.println(e);}
      }

      System.out.println(MyThread.count);
   }
}
```

Q4: How many threads in total exist for this program?

- 6, one created by each **start** call, one created by each **run** call.
- 7, one created by each **start** call, one created by each **run** call, and the main thread that the program initially runs in.
- 3, one created by each **start** call.
- 4, one created by each **start** call and the main thread that the program initially runs in.

Q5: Why is the statement *threads[i].join( );* in the *Main* method?

a. No reason other than to confuse the students and make this question possible.

b. To ensure all *MyThread* threads finish before printing out the value of *MyThread.count*.

Constructor call order question.

```
class MyBase {

   public MyBase( ) {
      System.out.println("0 arg MyBase");
   }

   public MyBase(String s) {
      System.out.println(s);
   }
}

class MyObject extends MyBase {

   public MyObject( ) {
      System.out.println("0 arg MyObject");
   }

   public MyObject(String s) {
      super(s+" base");
      System.out.println(s);
   }
}
```

```
class Main {

   public static void main(String args[]) {

      MyObject o1 = new MyObject( );
      MyObject o3 = new MyObject("o3");

   }
}
```

Q6: What is printed by the program?

a.
0 arg MyBase
0 arg MyObject
o3 base
o3

b.
o3 base
o3

c.
0 arg MyObject
0 arg MyBase
o3
o3 base

d.
o3
o3 base

e.
0 arg MyObject
o3
o3 base

f.
0 arg MyObject
o3 base
o3

Method call resolution and overloading questions. This program may contain intentional errors related to questions below.

```
class B {

   public char c = 'B';

   public void callPrint( ) {
      print( );
   }

   public void print( ) {
      System.out.println("bv");
   }

   public void print(int i) {
      System.out.println("bi");
   }

   public void print(int i, float f) {
      System.out.println("bif");
   }
}

class D extends B {

   public char c = 'D';

   public void print( ) {
      System.out.println("dv");
   }

   public void print(short s) {
      System.out.println("ds");
   }

   public void print(short s, double d) {
      System.out.println("dsd");
   }
}
```

```
class Main {

   public static void main(String args[]) {

      B b = new B( );
      D d = new D( );

      d.print((short) 4, (float) 4.0); // Q7
      d.callPrint( ); // Q8
      b.callPrint( ); // Q9

      b = d;
      b.print((short) 2); // Q10
      d.print((short) 2); // Q11
      b.print((int) 32); // Q12
      d.print((int) 32); // Q13
      System.out.println(b.c); // Q14
      System.out.println(d.c); // Q15
   }
}
```

Q7 – Q15: What is printed by each line in *main*? The question number is given in the comment for the statement. If the line is in error answer "E".

Static methods and class fields and final methods and fields. This program may contain intentional errors related to questions below.

```
class B {
   public static final double pi = 3.14;

   public void callPrint( ) {
      print( );
   }

   public static void print( ) {
      System.out.println("bv");
   }

   public static final void print(int i) {
      System.out.println("bi");
   }

   public static void print(double d) {
      System.out.println("bd");
   }
}

class D extends B {

   public static void print( ) {
      System.out.println("dv");
   }

   public static void print(int i) {
      System.out.println("di");
   }
}
```

```
class Main {

   public static void main(String args[]) {

      B b = new B( );
      D d = new D( );

      B.pi = 3.1415;
      b.print( ); // Q18
      d.print( ); // Q19
      b.print(32); // Q20
      d.print(4.0); // Q21
      d.callPrint( ); // Q22

      b = d;
      b.print( );
      b.print(32); // Q23
      b.callPrint( ); // Q24
   }
}
```

Q16: What is the effect of the statement *B.pi = 3.1415;* in main?

a. Assigns *3.1415* to *B.pi*.

b. An error because we cannot read or write variables declared within a class outside of the class.

c. An error because *B.pi* is declared *final* and we are reassigning it.

d. An error because *B.pi* is declared *static* and we are reassigning it.

Q17: Pick the most correct statement.

a. Each *B* object has its own copy of *pi*.

b. Because *pi* is declared *static* there is one copy of *pi* that is shared by every *B* object.

c. Because *pi* is declared *final* there is one copy of *pi* that is shared by every *B* object.

d. For there to be one copy of *pi* that is shared for every *B* object *pi* must be declared both *static* and *final*.

Q18 – Q24: What is printed by each line in the program that is labeled in a comment with a question number? If the line contains an error answer "E".

Abstract classes and interface questions. This program may contain intentional errors related to questions below.

```
interface I {

   int i = 4;

   public void print( ) {
      System.out.println("bv");
   }

   public void print(int i);

   public void print(int i, float f);
}

class D implements I  {

   public char c = 'D';

   public void print(int i) {
      System.out.println("pi");
   }

   public void print(int i, float f) {
      System.out.println("pif");
   }
}
```

```
class Key {

   public static void main(String args[]) {

      I i = new I( );
      D d = new D( );

      d.i = 5;   }
}
```

Q25 – Q29: Answer **true** (**T**) or **false** (**F**) for each of the following.

Q25. Java lets interfaces define methods (such as is done with *void print( )* in interface I) as a convenience to the programmer.

Q26. Java lets abstract classes define, as well as declare, methods (such as is done with *void print( )*.

Q27. Attempting to instantiate an interface, as is done in the statement *I i = new I( );* in *main* is allowed by Java as long as no abstract method is called.

Q28. Attempting to instantiate an interface, as is done in the statement *I i = new I( );* in *main*, is prohibited by Java.

Q29. The statement *d.i = 5;* in *main* is allowed by Java as long as *i* is not explicitly declared *final* or *private*.

```
interface I {
   public void print( );
   public void print(int i);
}

abstract class A {
   int i = 4;
   public void print( ) {
      System.out.println("bv");
   }
   public abstract void print(int i);
   public abstract void print(int i, float f);
}
```

```
class D extends A  {

   public char c = 'D';

   public void print(int i) {
      System.out.println("pi");
   }
   public void print(int i, float f) {
      System.out.println("pif");
   }
}

class Key {

   public static void main(String args[]) {

      A a = new A( );
      D d = new D( );
   }
}
```

Q30 – 35: Answer **true** (**T**) or **false** (**F**) for each of the following.

Q30. Class D does not have to define, as well as declare, methods *void print(int i);* and *void print(int i, float f)* as long as they are not called.

Q31. That the interface I and abstract class A both declare the same abstract method name makes using them in *D* illegal.

Q32. Even though interface *I* and abstract class *A* both declare *void print( )* and *void print(int i)*, they only need to be defined once in D.

Q33. Abstract classes cannot be instantiated (such as in the statement *A a = new A( );*) because they contain abstract methods that are declared but not defined (e.g., *void print(int i);*.)

Q34. A Java class can extend multiple abstract classes as long as the abstract classes don't declare the same method.

Q35. A Java class can extend multiple interfaces.

ECE 30862 Fall 2014 Second Exam Answer Sheet

All answers should be on this sheet. Both this sheet and your test must be signed and turned in. You may detach this sheet from the rest of the test to make it easier to write your answers on it. The first five questions are worth 5 points and the rest are worth 2.5 points. Putting your name on this sheet is worth 2 points.

I promise that I have neither given nor received disallowed aid on this test.

Name (Printed):

Name (Signed):

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.