CS56—Midterm 2 (E02)

W15, Phill Conrad, UC Santa Barbara

Monday, 03/02/2015

Name: _____

Umail Address: _____@ umail.ucsb.edu

Circle one: **4pm** **5pm** **6pm**

Please write your name **only** on this page.
That allows me to grade your exams without knowing whose exam I am grading.

This exam is **closed book, closed notes**, **closed mouth, cell phone off**,
except for:

- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes
- These sheets will be collected with the exam, and might not be returned
- Please write your name on your notes sheet

There are 100 points worth of questions on the exam, and you have 75 minutes (1 hr 15 minutes) to complete the exam.

A hint for allocating your time—on your first pass through the exam:

- if a question is worth 4 points, spend no more than 2 minutes on it
- if a question is worth 10 points, spend no more than 5 minutes on it
- etc.

If you do that, after you complete your first pass through the exam in 50 minutes, you'll still have 25 minutes to:

- revisit any questions where you need more time
- check your work.

Name: _____ **Handout**

1. (10 pts) (Short answer) Explain the difference between == and .equals() in Java.

   For full credit: include a specific explanation, as well as **specific code examples** in your explanation that **clearly** illustrate the difference.

3 Name: _____ Handout

The first three questions on this exam refer to the file Person.java, which appears on a separate handout along with this exam.

2. (4 pts) Consider the variable `name` as it appears on line 9. Which statement is *completely* true about this variable?
(Note: Some of the statements may be partially true, and partially false, while others are completely false. Only one is completely true.)

(**Multiple choice: circle one letter**)

a. It is a reference variable, and it is also an instance variable. So the reference is stored on the heap.
b. It is a reference variable, and it is a formal parameter. So the reference itself is stored on the stack.
c. It is a primitive variable, and it is also an instance variable. So the value is stored on the heap.
d. It is a primitive variable, and it is a formal parameter. So the value is stored on the heap.
e. It is a reference variable, and it is also a primitive variable. So the value is stored on the stack.

3. (4 pts) Consider the variable `name` as it appears on line 13. Which statement is *completely* true about this variable?
(Note: Some of the statements may be partially true, and partially false, while others are completely false. Only one is completely true.)

(**Multiple choice: circle one letter**)

a. It is a primitive variable, and it is also an instance variable. So the value is stored on the heap.
b. It is a reference variable, and it is also an instance variable. So the reference is stored on the heap.
c. It is a reference variable, and it is also a primitive variable. So the value is stored on the stack.
d. It is a primitive variable, and it is a formal parameter. So the value is stored on the heap.
e. It is a reference variable, and it is a local variable. So the reference itself is stored on the stack.

4. (4 pts) Consider the variable `x` as it appears on line 21. Which statement is *completely* true about this variable?
(Note: Some of the statements may be partially true, and partially false, while others are completely false. Only one is completely true.)

(**Multiple choice: circle one letter**)

a. It is a reference variable, and it is also a primitive variable. So the reference is stored on the heap,
But the value referred to is stored on the stack.
b. It is a reference variable, and it is also an instance variable. So the reference is stored on the heap,
and the value referred to is also on the heap.
c. It is a primitive variable, and it is a formal parameter. So the value is stored on the heap.
d. It is a primitive variable, and it is also an instance variable. So the value is stored on the heap.
e. It is a reference variable, and it is a local variable. So the reference itself is stored on the stack,
but the object it refers to is on the heap.

The next two questions refer to the following code listing

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <project default="compile">
3
4    <target name="compile" >
5      <mkdir dir="build" />
6      <javac srcdir="src" destdir="build" debug="true" debuglevel="lines,source">
7       <include name="**/*.java"/>
8      </javac>
9    </target>
10
11   <target name="run" depends="compile" >
12     <java classname="foo" fork="true" classpath="build" />
13   </target>
14
15 </project>
```

5. (4 pts) Which of the following is true about this code?

a. It is an example of a git commit metatdata file in XML format.
b. It is an example of an XML file used to configure a SQL database.
c. It is an example of a build.xml file used by Ant
d. It is an example of an XML file used by a servlet container (e.g. Apache Tomcat, Google App Engine)
e. It is an example of an HTML file to create a JSP (Java Server Page)

6. (4 pts) Consider the word destdir on line 6. What best describes this word?

a. It is a self closing element
b. It is the name of an attribute
c. It is the value of an attribute

d. It is an open tag
e. It is a nested element

The next few questions do not depend on the code on any handout.
Except where noted, consider each one as a stand-alone question.

7. (4 pts) What does it mean if a class implements the Serializable interface? (Multiple choice, circle one letter):
   a. It marks the object for expedited deletion by the Garbage Collector (through SerialDeletion), thus improving memory performance in programs that create large numbers of objects.
   b. Objects have serial numbers that can be used in place of references to look up objects—the serial number is accessed through the .serial() method of java.lang.Object
   c. It means that instances of that class inherit methods from the base class Serializable that allow them to be streamed to a file, or over a network, or restored from a file or a network connection.
   d. The class implements methods that allow Object instances can be written to an ObjectOutputStream and read from an ObjectInputStream—in this way, they can be stored to a file, and read from a file, or sent over a network connection from one process to another.
   e. Objects can be put into a strict order (by serial number) and sorted when they occur in Java Collections such as Lists, Sets, ArrayLists, etc.
8. (4 pts) Suppose the first line of a class is `public class Foo implements Runnable`.
   Which of the following is a completely correct statement about the `Foo` class?
   a. Because `Foo` implements `Runnable`, `Foo` instances can be run directly from an Ant `<runnable>` task.
   b. Because `Foo` implements `Runnable`, `Foo` the author of the Foo class has the obligation to write a `public void run() {}` method that is called when instances of the `Foo` class are used to create a new `Thread`
   c. Because `Foo` implements `Runnable`, it can never be instantiated as an object—it can only be "run" by the JVM.
   d. Because `Foo` implements `Runnable`, any thread created from am instance of Foo is always in the RUNNABLE state—never in the NEW, BLOCKED, WAITING, TIMED_WAITING, or TERMINATED states.
   e. Because `Foo` implements `Runnable`, `Foo` inherits a `public void run()` method from `Runnable` that can be overridden to make `Foo` a runnable thread.
9. (4 pts) Consider the following code:

```
try {

   Thread.sleep(2000);

} catch(InterruptedException ex) {

    ex.printStackTrace();

}
```

   Assuming there is no interruption that causes an Exception, when the 2000 milliseconds of sleep is over, into what state will this thread go?

   a. awake
   b. runnable
   c. new
   d. blocked
   e. running
10. (4 pts) Assume the following code appears inside a main() method of a Java class.

```
int x = 1;

while (x) {

   System.out.println("foo");
   x++;

}
```

   Which of the following is true about this code?
   a. It will print "foo" on standard output, one "foo" per line, until the value of x exceeds the maximum value for an int (i.e. $2^{32} - 1$), at which point an Exception will be thrown.
   b. It will print "foo" on standard output, one "foo" per line, until the value of x wraps around to the largest negative integer, i.e. -2^{32}, and then climbs back up to zero. At that point, the condition will be false and the loop will halt.
   c. It will print "foo" on standard output, one "foo" per line, forever, or until the program is terminated
   d. It is a syntax error—int values cannot be treated as boolean values in Java

11. (4 pts) Assume that the following line of code appears inside a java main() method.

```
Dog [] pets = new Dog[7];
```

Which of the following is a true statement?

a. `pets` is a local variable on the stack that refers to an Array object. That array object is on the heap. The array object contains seven Dog references pets[0] through pets[6], each of which is initially null. No Dog objects are created by this statement.

b. `pets` is a local variable on the stack that refers to an Array object. This is an array of seven Dog objects, pets[0] through pets[6] each of which is allocated on the heap. This statement is not legal if the Dog class does not contain a default constructor.

12. (4 pts) Which of the following is true about normal git workflow:
    a. The normal sequence is "git add", "git push", "git commit"
    b. The normal sequence is "git commit", "git add", "git push"
    c. The normal sequence is "git add", "git commit", "git push"
    d. The normal sequence is "git push", "git commit", "git add"
    e. The normal sequence is "git push", "git add", "git commit"
13. True or False

(2 pts)	If Geegaw is a class that contains all of the methods of the Foobarable interface, then the correct syntax for this is `... class Geegaw extends Foobarable...`	T	F
(2 pts)	On a particular specific JVM, the amount of memory taken up for an object **reference** can differ for different kinds of objects (say String vs. ArrayList<String>, vs. JPanel.	T	F
(2 pts)	A class can extend multiple classes, but can implement only one interface	T	F
(2 pts)	On a particular specific JVM, the amount of memory taken up for an object **instance** can differ for different kinds of objects (say String vs. ArrayList<String>, vs. JPanel.	T	F
(2 pts)	In Java, one can create an array of primitive values, an array of objects, or an array of object references.	T	F
(2 pts)	An Java array can automatically grow to the size needed—it is not necessary to specify an initial size for the array when creating the array instance.	T	F
(2 pts)	If you have an abstract class called Foo, it is not legal in Java to create a reference of type Foo, i.e. `Foo f;` would be a syntax error.	T	F

14. A major feature of Java is that it has automatic garbage collection. BRIEFLY EXPLAIN:

    a. (5 pts) What is automatic garbage collection in a programming language such as Java?

    b. (5 pts) What is the benefit of automatic garbage collection to a development team working in Java?

15. Another major feature of Java is that classes can be marked as "serializable". BRIEFLY EXPLAIN:

    a. (4 pts) What do you write in the code to mark a class as serializable?

    b. (4 pts) What capability do you get when you indicate that a class is serializable?

    c. (4 pts) You can put the keyword `transient` in front of an instance variable in a class that is serializable (e.g. `transient String foo;`). What does it mean when you see this keyword `transient`?

16. Write a line or two of code that does "auto-boxing" involving the primitive and wrapper types used for integers in Java, and THEN write those same lines of code the way a programmer would have to have written it BEFORE autoboxing was added to the Java language. As a hint: the class `java.util.ArrayList<E>` has methods:
    `void add(int index, E element)` and
    `E get(int index)`

    a. (5 pts) The line of code that does auto-boxing:
    CAUTION: be sure your code does only auto-boxing, and NO auto-unboxing.
    If you need an extra line of code or two to set up your auto-boxing code (e.g declaring variable, etc.), that's ok.

    b. (5 pts) The SAME lines of code written WITHOUT any autoboxing.
    CAUTION: be sure your code does auto-boxing, not auto-unboxing.
    If you need an extra line of code or two to set up your auto-boxing code (e.g declaring variable, etc.), that's ok.

End of Exam

Total Points: 100

Handout for CS56, W15 Midterm E02

Person.java

Person.java

```
1  public class Person {
2      private String name;
3      private int age;
4
5      public Person(String name, int age) {
6          this.name = name; this.age = age;
7      }
8
9      public String getName() { return name; }
10     public int getAge() { return age; }
11
12     public String toString () {
13         return "(" + name + "," + age + ")";
14     }
15
16     public static void main(String [] args) {
17         String name = "Fred";
18         int age=32;
19
20         Person y = new Person(name,age);
21         Person x = new Person("Wilma",31);
22
23         System.out.println("A: x = " + x + " y = " + y);
24
25         name = "Betty";
26         age = 29;
27         x = new Person(name,age);
28         y = new Person("Barney",29);
29
30         System.out.println("B: x = " + x + " y = " + y);
31     }
32
33 }
```