| Name: *(as it would appear on official course roster)* |
|---|
| Umail address: @umail.ucsb.edu |

1

# EXAM: e01: Midterm Exam

e01

CS56 F16

| ready? | date | points |
|---|---|---|
| true | Tue 10/18 12:30PM | 100 |

*You may not collaborate on this exam with anyone. If you need to use the restroom, you must leave your cell phone with the exam proctor before leaving the room.*

- Write your name at the top of this page **AND EVERY ODD NUMBERED PAGE.**
- Double check that you turned in ALL pages; look for "End of Exam" on the last page.
- This exam is **closed book, closed notes**, **closed mouth, cell phone off.**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes.
- This sheet will be collected with the exam, and might not be returned.
- Please write your name on your notes sheet.

---

1. (10 pts) On the handout there is some code. Your job: figure out after which line of main() each of the following objects is eligible for garbage collection.

   If an object is still not eligible for garbage collection when the last line of main is reached, write “never”. Each answer should be a line number, or the word never.

| Object | Fill in line here |
|---|---|
| (a) `Fido` | |
| (b) `Princess` | |
| (c) `Rover` | |
| (d) `Snoopy` | |
| (e) `Spot` | |

2. (5 pts) On the handout you got for this exam, there is a `Dog` class. The `Dog` class, as written, lacks an overridden `toString` method.

   Write the code for an overridden `toString` method the returns the Dog’s name.

3. (10 pts) Assume that in a given program, there might be a variable `names` that is of type `ArrayList<String>`, and contains an unknown number of dog names. For example, it might contain `Fido`, `Ralph`, and `Snoopy`.

   Write a static method that we could add to the `Dog` class that takes such a variable as its parameter, and returns an `ArrayList<Dog>` containing `Dog` object with precisely those names.

   Hint: Be very careful about the difference between how `add` and `set` operate on `ArrayList` objects.

# 2

# e01

## CS56 F16

4. (10 pts) Suppose some constructor method is declared in its Javadoc to be "known to throw" some kind of checked exception—for example,

```
public FileReader(File file)
            throws FileNotFoundException
```

   And you are writing a line of code such as this inside some method.

```
BufferedReader br = new BufferedReader(new FileReader(new File("file.txt")));
```

   Suppose your method is:

```
public static void foo() { ... }
```

   When you see that the javadoc says `throws FileNotFoundException` and you lookup and discover that `FileNotFoundException` is a checked exception, as programmer, you now have two choices. Briefly explain the two choices.

| | |
|---|---|
| **Name:**<br>*(as it would appear on official course roster)* | |
| **Umail address:** | **@umail.ucsb.edu** |

# 3

# e01

## CS56 F16

5. (5 pts) We are using a third-party library called JUnit in this course. If you were asked at a job interview to briefly describe the purpose of `JUnit`, what would you say?

   Include *enough detail* in your answer so that the interview knows that you are technically sharp, and they should hire you. Do not include so much extra detail that the interviewer finds you tedious and annoying, and decides you would be painful to work with, and chooses to not hire you.

6. (5 pts) Same question, but this time `ant` and the `build.xml` file.

   If you were asked at a job interview to briefly describe the purpose of `ant` and the ~~`buidl.xml`~~`build.xml` file, what would you say? Same comment as above about the "level of detail".

7. On the handout you got for this exam, there is a `Dog` class. The `Dog` class, as written, lacks an overridden `.equals` method.

   Yet, it is still possible to invoke `.equals()` on Dog objects.

   Briefly, but preicsely, answer these questions.

   a. (5 pts) Write two lines of code that declare two variables `a` and `b`, each of which is a Dog reference, and initialize both to separate Dog objects with the name `Alice`.

# 4

# e01

## CS56 F16

   b. (5 pts) Write a few lines of java that compare the objects referred to by `a` and `b` using the `.equals` method, and then print "EQUAL" on the standard output stream if the method returns true.

   c. (10 pts) Briefly **explain** how the `.equals()` method invoked in this case (with no overloaded `.equals` method in `Dog`) determines whether to return `true` or `false`, *both* in this specific case, *and* in general. (Both parts needed for full credit).

   d. (5 pts) Continuing with the `Dog` class: if/when you override the `.equals()` method for a class, it is considered a "Java best practice" to always override at least one other method—

      In below, write the "signature" for that method, i.e. the return type, name of method, and the parameters it takes if any (the same way it appears in Javadoc, for example.)

      Hint: if you are drawing a blank, check the method list for `java.util.ArrayList<E>`, because its on the list. That turns this into a "multiple choice question, albeit one with 42 choices.

| **Name:**<br>*(as it would appear on official course roster)* | |
|---|---|
| **Umail address:** | **@umail.ucsb.edu** |

# 5

# e01

## CS56 F16

8. (10 pts) Continuing with the `Dog` class from the handout: handout

   In the space below, please write a properly written .equals() method for Dog.

   The reverse side of the `ArrayList` handout has some reminders about a properly written `.equals()` method.

9. (10 pts) Continuing with the `Dog` class from the handout: handout

   In the space below, please write the full code for the other method you need to override when ever you override .equals(). (Hint: consider that java.lang.String already has this method.)

10. (10 pts) For each of the following indicate if the line of code involves auto-boxing, and/or auto-unboxing. If a line of code involves both, check both boxes. If it involves neither, check neither box. ASSUME THAT ALL THE LINES OF CODE ARE IN THE SAME `main` METHOD, CONSECUTIVELY.

(Grading: -2 for each incorrect answer, but no more than -10 total.)

| **Code** | **auto-boxing** | **auto-unboxing** |
|---|---|---|
| `ArrayList<Integer> mylist = new ArrayList<Integer>();` | ☐ | ☐ |
| `mylist.add(9);` | ☐ | ☐ |
| `mylist.add(new Integer(3));` | ☐ | ☐ |
| `mylist.add(mylist.get(0) + 1);` | ☐ | ☐ |
| `int x = mylist.get(0);` | ☐ | ☐ |
| `Integer y = mylist.get(1);` | ☐ | ☐ |

**End of Exam**

**6**

**e01**

**CS56 F16**