CSE1030Z Midterm test
Friday, March 1, 2013
10:30 AM–11:20 AM

This is a closed book test. No aids are permitted except for a non-electronic dictionary. Answer the questions in the spaces provided on the question sheets. You may use the back of the pages if you need more space for your answers.

Name and student number: _____

1. / 10

2. / 10

3. / 10

4. / 10

5. / 10

1. (10 points) Static and non-static attributes

   (a) (5 points) What is the main difference between a static attribute and a non-static attribute?

   Solution: There is only one copy of a static attribute and it belongs to the class. There is one copy of each non-static attribute for every object.

   (b) (5 points) Consider the following class:

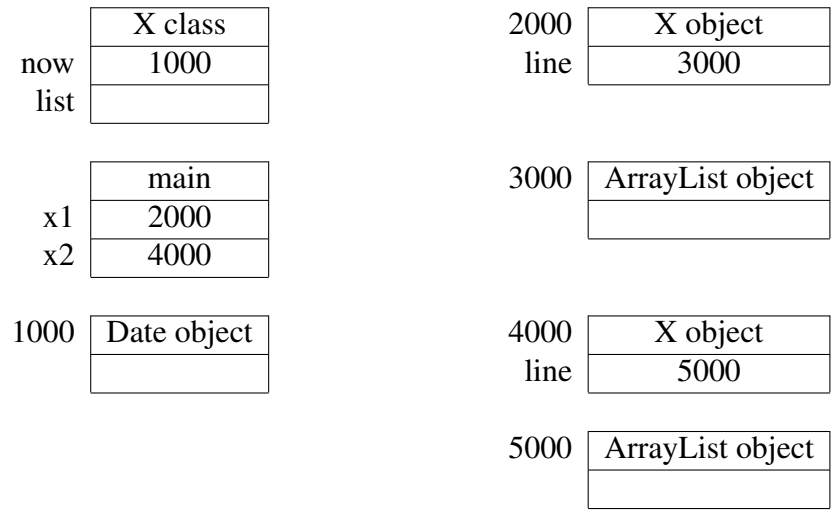
```
public class X {
  private static Date now = new Date();
  private List<String> list;

  public X() {
    this.list = new ArrayList<String>();
  }

  public static void main(String[] args) {
    X x1 = new X();
    X x2 = new X();
    // more code here...
  }
}
```

   Draw the memory diagram of the running program immediately after the second line of the `main` method has run. Your diagram should show the class `X`, the `main` method, and all objects that have been created.

   Solution: There should be 1 Date object, 2 X objects, and 2 ArrayList objects.



2. (10 points) Constructors

   (a) (5 points) What is the main purpose of the constructors for a class?

   Solution: To create and initialize the attributes of an object when it is instantiated.

   (b) (1 point) Suppose you have a class named `Foo`. What is the signature of the default constructor?

   Solution: Foo()

   (c) (1 point) Suppose you have a class named `Foo`. What is the signature of the copy constructor?

   Solution: Foo(Foo)

   (d) (1 point) What is meant by the term "constructor chaining"?

   Solution: When a constructor invokes another constructor of the same class or of the immediate super class.

   (e) (2 points) What is the main benefit of constructor chaining?

   Solution: Reduces duplication of code.

3. (10 points) Testing objects for equality is an important task in Java (and most other object oriented languages).

   (a) (5 points) For two object references `a` and `b` (of type `String` or `Date`, for example), what is the main difference between `a == b` and `a.equals(b)`?

   Solution:
   `a == b` checks if `a` and `b` are references to the same object;
   `a.equals(b)` checks if `a` and `b` have the same state

   (b) (1 point) The equals contract says that `equals` must be reflexive; explain what this means.

   Solution: `a.equals(a)` is always `true` (if `a` is not `null`)

   (c) (1 point) The equals contract says that `equals` must be symmetric; explain what this means.

   Solution: `a.equals(b) == b.equals(a)`

   (d) (3 points) List the main steps you need to perform when implementing `equals` for a class named `X`. The declaration for `equals` is:

   ```
   public boolean equals(Object obj)
   ```

   Step 1 is done for you:

   1. check if `this == obj`

   Solution: Only any 3 of the 4 following items were required:

   2. check if `obj == null`
   3. check if `this` and `obj` have the same class
   4. cast `obj`
   5. compare the attributes of `this` and `obj`

4. (10 points) Comparable

   (a) (4 points) Suppose that a class represents objects with a natural ordering (such as strings, dates, and playing cards). What is the advantage of implementing the Comparable interface for such classes?

   Solution: We can easily sort collections of objects (using `Collections.sort`) and we can easily use the tree-based collections (`TreeList` and `TreeMap`).

   (b) (2 points) What does it mean if `compareTo` is consistent with `equals`?

   Solution: Technically, both conditions below are required, but either was accepted for full marks.

   if `x.compareTo(y) == 0` then `x.equals(y) == true`
   and if `x.equals(y) == true` then `x.compareTo(y) == 0`

   Note that `compareTo` and `equals` do not return the same result: `compareTo` returns an integer and `equals` returns a boolean value.

   (c) (2 points) Give an example of a class where it is sensible that `compareTo` is *not* consistent with `equals`

   Solution: Some examples:

   Playing cards: The usual sorting by value compares only the ranks, whereas equality usually compares both the rank and suit. Lab 3 was unusual in that both the rank and suit was used in `compareTo`.

   `Event` from the lab test: `compareTo` returns `0` if events overlap, whereas `equals` returns true if and only if the start and end times are the same.

   `Line` from the lab test: `compareTo` returns `0` if lines have the same length, whereas `equals` returns true if and only if the start and end points are the same.

   (d) (2 points) Consider how one normally implements `equals` and `compareTo`. In general, when will `compareTo` be consistent with `equals` and when will `compareTo` be inconsistent with `equals`?

   Solution: If `compareTo` and `equals` consider different attributes then `compareTo` be will usually be inconsistent with `equals`; e.g., ranks versus ranks and suits for playing cards, length versus start and end point for lines.

5. (10 points) Aggregation and composition

(a) (2 points) Aggregation and composition both model what relationship between two classes?

Solution: has-a

(b) (2 points) What is the main difference between aggregation and composition?

Solution: Composition imples shared lifetime or ownership and aggregation does not.

(c) (1 point) Is is often said that immutable objects make great building blocks for other objects; explain why this statement is true in the context of aggregation and composition.

Solution: Defensive copies of immutable objects are not needed because their state cannot change.

(d) (2 points) When implementing a composition, we often have to make defensive copies to prevent privacy leaks. Explain what a privacy leak is and why it can be harmful.

Solution: A privacy leak is when a client obtains a reference to a private mutable attribute inside another object. This allows the client to arbitrarily change the state of the attribute without asking the object.

(e) (3 points) Suppose that `t` is a list of 7 `Date` objects. Fill in the table below indicating the number of new list objects and new `Date` objects created after making an alias to `t`, a shallow copy of `t`, and a deep copy of `t`.

	number of new list objects	number of new `Date` objects
alias	0	0
shallow copy	1	0
deep copy	1	7