This is a CLOSED–BOOK quiz.

Before you start, write your name at the top of every sheet.

There are 6 parts, labeled A through F. Please check your copy of the quiz before you start to make sure it is complete: you should have 5 sheets, printed on 10 sides in total. Note that the parts have unequal value.

You have 75 minutes and should attempt to answer all questions. Good luck!

<table>
<thead>
<tr>
<th>Part</th>
<th>Score</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Hashing and Invariants

Ben Bitdiddle is building a user registration mechanism for a website, and has implemented an abstract data type User as a Java class. He has included a method make that he intends clients of the data type to use in place of the constructor. This method is designed to establish the invariant that no two distinct users have the same id; to do this, it stores all previously generated User objects in the static set issued and compares a newly generated one to these with an equals method that matches on the id field.

1. What pattern is Ben using in the method make? (Choose one.)

a. Interning
b. Factory
c. Visitor
d. Iterator

Ben is having trouble getting his program to work, and he asks Alyssa Hacker for advice. Fortunately, she’s taking 6.005 and she immediately spots several mistakes in his code. Unfortunately, she rushes
off to another class, leaving Ben to figure out how these mistakes might be causing the problems he’s observing.

2. One problem Ben is observing is that his program gets slower and slower after running for long periods of time. He suspects some kind of memory management problem. Which of the following mistakes that Alyssa noticed could be responsible for this? (Select Yes or No for each.)

a. Y N Creating a new User object in make is wasteful, because memory is allocated even if the object is rejected.

b. Y N The issued set should hold ids as strings, and not whole User objects.

c. Y N The initialization of issued should have used the HashSet constructor that takes a size hint, so that the first hash set object allocated could accommodate the many users of Ben’s website, without the need for repeated resizing.

Another problem Ben has is that distinct users are ending up with matching ids, undermining his intended invariant. From some debugging, he has discovered that

A. A User may be returned by make with an id that has already been issued by make before.

B. A User may be returned by make with the id of another existing user.

C. Two User objects may come to have the same id even though they did not have the same id initially.

3. For each of the following mistakes that Alyssa found, circle the strongest observation that might be caused by that mistake alone (where A is the strongest and C is the weakest), or none if the mistake has none of these effects. For example, if you think that the misspelling would cause observations A and B, you would circle A for part a.

a. A B C none The name of the hashing method is misspelled; it should be hashCode.

b. A B C none Ben forgot to replace the body of the hashing method with the real computation, and left in the dummy code that just returns a constant.

c. A B C none The equals method has the wrong type; it should take an Object, not a User.

d. A B C none The constructor should be private.

e. A B C none The User class should be immutable, and shouldn’t have a mutator such as changeId.

f. A B C none By returning null instead of throwing an exception, the make method allows a client to continue computing with a bad return value.
B  Equality Properties

Study the equals methods of these two classes:

```java
public class Account {
    private double balance;
    ...

    public boolean equals (Object obj) {
        if (! (obj instanceof Account)) return false;
        Account a = (Account) obj;
        return this.balance == a.balance;
    }
}

public class InterestBearingAccount extends Account {
    private double interestRate;
    ...

    public boolean equals (Object obj) {
        if (! (obj instanceof InterestBearingAccount))
            return false;
        InterestBearingAccount a = (InterestBearingAccount) obj;
        return this.balance == a.balance && this.interestRate == a.interestRate;
    }
}
```

1. The method InterestBearingAccount.equals: (Choose one.)
   a. Is correct (that is, satisfies the Object Contract).
   b. Is incorrect, since it violates the transitivity requirement.
   c. Is incorrect since it is non-deterministic.
   d. Is incorrect, since it violates the symmetry requirement.

Now suppose the code of this method is replaced by

```java
public boolean equals (Object obj) {
    if (! (obj instanceof InterestBearingAccount))
        return super.equals(obj);
    InterestBearingAccount a = (InterestBearingAccount) obj;
    return this.balance == a.balance && this.interestRate == a.interestRate;
}
```

2. This implementation: (Choose one.)
   a. Is correct (that is, satisfies the Object Contract).
   b. Is incorrect, since it violates the transitivity requirement.
   c. Is incorrect since it is non-deterministic.
   d. Is incorrect, since it violates the symmetry requirement.
C  Java Semantics

1. What is printed as a result of executing the following code?

```java
String a = "rock";
String b = a;
a.concat(" paper");
b.concat(" scissors");
System.out.println(a);
System.out.println(b);
```

a. rock paper
   rock paper scissors
b. paper
   scissors
c. rock paper
   rock scissors
d. rock
   rock

2. Which literal objects are reachable after execution of this segment? (Circle Yes or No.)

a. Y N rock
b. Y N paper
c. Y N scissors
3. What is printed as a result of executing the following code? (Select one.)

```java
List<String> friends = new LinkedList<String>();
List<String> enemies = new LinkedList<String>(friends);
List<String> opponents = enemies;
friends.add("Alice");
friends.add("Bob");
enemies.add("Carol");
opponents.add("Dave");
for (String str : friends) System.out.print(str + " ");
System.out.println();
for (String str : enemies) System.out.print(str + " ");
System.out.println();
for (String str : opponents) System.out.print(str + " ");
System.out.println();
```

- a. Alice Bob Carol
- b. Alice Bob Carol Dave
- c. Alice Bob Carol Dave
- d. Alice Bob Carol Dave

4. Which of the following types are mutable in Java? (Circle Yes or No for each.)

- a. Y N float[]
- b. Y N String
- c. Y N List<String>
- d. Y N int
- e. Y N MyImmutableSet<List<String>>
D Listener Patterns

1. A program that uses Swing has: (Choose one.)
   a. just one Swing thread and any additional threads started explicitly by the programmer
   b. at least two threads: one Swing thread and the main thread
   c. a Swing thread for each container
   d. a Swing thread for each listener

2. Having model code make explicit calls to Swing objects: (Choose one.)
   a. is good because it’s the easiest way to program updates
   b. is good because it improves performance
   c. is bad because it can result in data races

3. When a thread calls invokeLater with an event as argument: (Choose one.)
   a. the thread itself executes that event later
   b. a new thread is spawned that executes that event later
   c. the Swing thread executes that event later

4. A well designed Swing program does not suffer from data races in the UI because: (Choose one.)
   a. all Swing objects are thread-safe
   b. Swing objects cannot be accessed outside the Swing framework
   c. mutators are called on Swing objects only by a single thread

5. The purpose of the Swing event queue is: (Choose one.)
   a. to allow important events to be prioritized
   b. to eliminate concurrency problems by allowing events to be executed sequentially
   c. to support buffering when multiple requests are issued rapidly

6. The pattern in which listeners are registered with objects: (Choose one.)
   a. reduces the dependence of objects on their listeners
   b. reduces the dependence of listeners on their objects
   c. reduces dependences in both directions
7. The key decoupling achieved by the MVC pattern is that: (Choose one.)
   a. the view is not dependent on the model
   b. the model is not dependent on the view
   c. the view is not dependent on the controller

8. Listeners attached to GUI objects: (Choose one.)
   a. should not spawn threads
   b. should not take a long time to execute
   c. should not cause mutations
E  Object Model Design

The designer of a drawing program has produced the object model below. Drawing objects can be organized hierarchically; the pair o1->o2 is in the relation elements if o2 is an element of o1. Each object has a bounding box; the pair o->c is in the relation boundingBox if the bounding box for object o includes coordinate c.

1. Which of the following invariants follow from the model? (Circle True or False for each.)
   a. T F The only shapes possible are ovals and rectangles.
   b. T F The only objects possible are shapes and groups.
   c. T F No shape is also a group.
   d. T F No shape is both a rectangle and an oval.
   e. T F Every shape is an object.
   f. T F The set Object is empty.
   g. T F A group cannot contain itself as an element.
   h. T F A shape cannot contain itself as an element.

2. Which pattern is used in this object model? (Select one.)
   a. Interpreter
   b. Composite
   c. Classifier

3. The object is not complete, because it lacks multiplicity symbols. Complete it by adding them, using your common sense and familiarity with drawing applications.
F Object Model Implementation

1. You are designing an online forum, and have constructed the object model shown below. The relation replies contains the pair \( m1 \rightarrow m2 \) when \( m2 \) is one of the messages replying to \( m1 \). Using common sense and your familiarity with online forums, complete the object model by adding multiplicities.

Suppose the object model is to be implemented as a Java program, and you have decided to implement one class for each set in the object model, and to assign each of the relations in the object model to one of the classes (as a simple field or a set or a map). How you do this will depend on your design goals.

2. If making as many classes as possible immutable is your goal, how would you assign relations to classes? For each relation, circle the class it should be assigned to.
   a. members is assigned to: Date Forum Message Subject User
   b. messages is assigned to: Date Forum Message Subject User
   c. posted is assigned to: Date Forum Message Subject User
   d. subject is assigned to: Date Forum Message Subject User
   e. date is assigned to: Date Forum Message Subject User
   f. replies is assigned to: Date Forum Message Subject User

Suppose you gave yourself the freedom to represent as a field the transpose of a relation rather than the relation itself. For example, instead of representing members as a field of Forum, connecting a forum to its members, you could instead represent it as a field of User, connecting a member to the forums the member belongs to.

3. With the same immutability goal, but with this new freedom, what would you do now? For each relation, circle the class it or its transpose should be assigned to.
   a. members is assigned to: Date Forum Message Subject User
   b. messages is assigned to: Date Forum Message Subject User
   c. posted is assigned to: Date Forum Message Subject User
   d. subject is assigned to: Date Forum Message Subject User
   e. date is assigned to: Date Forum Message Subject User
   f. replies is assigned to: Date Forum Message Subject User