Midterm Test Solutions — Horton’s section

Duration: 50 minutes
Aids allowed: None

Make sure that your examination booklet has 6 pages (not including this one). Write your answers in the spaces provided. Write legibly.

Surname: __________________________  First name: __________________________

Student #: __________________________

Tutor (circle one):
A–C  D–Kl  Km–Me  Mf–Si  Sj–Z
Jennifer Listgarten  Irum Godil  Sarah Imrisek  Akil Sadikali  Shannon Dalmao

1. __________  / 3
2. __________  / 8
3. __________  / 8
4. __________  / 10
5. __________  / 4
Total __________  / 33
Question 1 [3 marks in total]

Suppose that the following silly little program compiles and runs:

```java
class EnumDriver {
    // A method you've seen in lecture.
    public static void printEnumeration( Enumeration i) {
        while (i.hasMoreElements()) {
            System.out.println(i.nextElement());
        }
    }

    public static void main(String[] args) {
        A a = new A();
        // Perhaps do some things to 'a' here, such as inserting things into it.
        printEnumeration(a);
        B b = new B();
        // Perhaps do some things to 'b' here, such as inserting things into it.
        printEnumeration(b.blah());
    }
}
```

(a) What do you know about class A?
Class A implements the interface Enumeration or some interface that is a descendent of Enumeration. (Otherwise, we could not pass an instance of A to method printEnumeration).

(b) What do you know about class B?
Class B has a method called blah with return type Enumeration, or some descendent interface of Enumeration. (Otherwise, we couldn't pass the result of the call "b.blah()" to method printEnumeration.)

Question 2 [8 marks in total]

Here is a class for defining nodes in a linked list:

```java
class Node{
    public Object data;
    public Node link;
}
```

Suppose we want a method that moves the first n nodes of a linked list to the end. For example, below is what it would do if n were 3, given a linked list of with 5 nodes. (The data field of each node is labelled with a letter in the diagram.)

```
Before After 3 nodes have been moved to the end
front
a b c d e
front
d e a b c
```

CONTINUED
Below is an outline for this method. Complete it. Don’t forget to fill in a return value.

Note: This code presumes also that \( n \leq \) the number of nodes in the linked list, which should have been a precondition.

```java
// Taken the first n nodes of a linked list and move them to the end.
// 'front' refers to the front node in the linked list.
// Return a pointer to the first node in the revised linked list.
// Precondition: \( n \geq 0 \).
public static Node moveToEnd (Node front, int n) {

    // Will point to the end of the revised linked list.
    Node newEnd = front;

    // Advance newEnd n-1 times from node 1, in order to reach node n.
    // Node newEnd will be the end node of the revised linked list.
    for (int i=1; i<= n-1; i++){
        newEnd = newEnd.link;
    }

    // [This if-statement could be moved up to include the code above --
    // and that would read better. But since the starter code didn’t
    // include the if, it’s fine to put it here.]

    // If n is zero, there is no work to do. Otherwise...
    if (n > 0) {

        // Will point to the end of the current linked list.
        Node end = newEnd;
        // Go find the end.
        while (end.link != null)
            end = end.link;

        // Update the links.
        end.link = front;
        front = newEnd.link;
        newEnd.link = null;
    }

    // [This return has to be outside the if, because even if we change
    // nothing, we are still obliged to have a return value.]

    // Return a reference to the front of the (possibly) revised linked list.
    return front;
}
```

CONTINUED
Question 3  [8 marks in total]

Below is a class that counts integers within a range of values.

class NumberCounter {
    private int n;
    private int[] count;
    private int numNums;
    private int total;

    public NumberCounter(int n) {
        this.n = n;
        count = new int[n];
        for (int i=0; i<n; i++){
            count[i] = 0;
        }
        numNums = 0;
        total = 0;
    }

    public void tally(int i){
        count[i]++;
        total = total + i;
        numNums++;
    }

    public int report(int i){
        return count[i];
    }

    public double average(){
        return ((double)total) / numNums;
    }

    public int biggestICanCount(){
        return n;
    }
}

(a) Write a representation invariant for this class:

- \( n \geq 1. \)
- \( \text{count.length} = n. \)
- \( \text{count[i]} \) is the number of times that \( i \) has been tallied (for all \( i \) from 0 to \( \text{count.length} \) inclusive). \( \text{count[i]} \geq 0 \) (for all \( i \) from 0 to \( \text{count.length} \) inclusive).
- \( \text{numNums} \) is the number of times \( \text{tally} \) has been called (or equivalently, the number of numbers that have been tallied).
  \( \text{numNums} \geq 0. \)
- \( \text{total} \) is the total value of all the numbers that have been tallied (or equivalently, \( \sum_{i=0}^{n-1} \text{count}[i] \times i \)).
  \( \text{total} \geq 0. \)

(b) Write an external comment for method \text{report}:

- Returns the number of times that \( i \) has been tallied.
- Precondition: \( 0 \leq i \leq \text{biggestICanCount()} \) (which is the upper range of this NumberCounter.)
- This answer should not mention any instance variables or reveal anything about the implementation of the class.
**Question 4** [10 marks in total]

This program compiles without errors.

```java
class Top {
    public int n = 1;
    public void report() {
        System.out.println("hi");
    }
}
class Middle extends Top {
    public int n = 2;
    public void report() {
        System.out.println("maybe");
    }
}
class Bottom extends Middle {
    public int n = 3;
    public void report() {
        System.out.println("bye");
    }
}
class Tester{
    public static void main (String[] args){
        Top a = new Bottom();  // Notice that it says "new Bottom" not "new Top".
    }
}
```

For each of the following questions, circle “Yes” or “No”, and explain your answer as requested.

(a) Inside **Tester’s main** method, can we access **Top’s report** method using variable `a`?

    Yes, and here’s the simplest way:
    
    ![No](http://example.com/no.png)
    
    and here’s why not: **Because report is a method.**

(b) Inside **Tester’s main** method, can we access **Bottom’s report** method using variable `a`?

    ![Yes](http://example.com/yes.png)
    
    and here’s the simplest way: `a.report()`.

    No, and here’s why not:

(c) Inside **Middle’s** `report` method, can we access **Top’s report** method?

    ![Yes](http://example.com/yes.png)
    
    and here’s the simplest way: `super.report()`;

    No, and here’s why not:

(d) Inside **Tester’s main** method, can we access **Top’s variable n** using variable `a`?

    ![Yes](http://example.com/yes.png)
    
    and here’s the simplest way: `a.n`

    No, and here’s why not:

(e) Inside **Tester’s main** method, can we access **Bottom’s variable n** using variable `a`?

    ![Yes](http://example.com/yes.png)
    
    and here’s the simplest way: `((Bottom)a).n`

    No, and here’s why not:

    **CONTINUED**
Additional remarks about the answers to question 4:

(a) We are trying to access a method, and Java’s rules for finding a target say to start looking in
the objects bottom box, regardless of the type of the expression used to access the method. (In
other words, casting will make no difference.) The bottom box inside the object referred to by a
is a Bottom, so we will find its report method. We can’t use super to go up to higher parts of
the object because super refers to the object whose method we are currently running, which is an
instance of Tester, not of Bottom.

(b) We don’t need to cast a to Bottom, even though the expression a’s type is Top, again because
of Java’s rule for finding a target when that target is a method.

(d) This expression gets Top’s variable without a cast because var a is declared a Top.

(e) We need a cast to get bottom’s variable, because var a is declared a Top.

Additional remarks about the answers to question 5:

The “Suzanne” line won’t compile because it is an attempt to access a protected member of a
class in another package, and yet doesn’t access it through inheritance. Instead, it tries to access it
directly in the class in which it is declared: other is a Person, and name is declared inside Person.
Question 5 [4 marks in total]

Here is a program that may have some problems.

```java
package A;
public class Person {
    protected String name;
    private int age;
    
    public void older(Person other){
        if (other.age > this.age)
            System.out.println("You are older than me.");
    }
}

package B;
import A.Person;
public class Student extends Person{
    
    public void tryPerson(Person other){
        name = "Toni";
        other.name = "Suzanne";
    }
    public void tryStudent(Student other){
        other.name = "Craig";
    }
}

package C;
import A.Person;
import B.Student;

class TestAccess{
    public static void main(String[] args){
        Student s1 = new Student();
        Student s2 = new Student();
        Person p = new Person();
        s1.tryPerson(p);
        s2.tryStudent(s2);
    }
}
```

For each of the following of lines out of the program, circle “OK” if there is nothing wrong with the line or “not OK” if it violates the rules of accessibility in Java. Do not guess; there is a penalty of 1 mark for wrong answers.

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Line</th>
<th>Your answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>older</td>
<td>if (other.age &gt; this.age)</td>
<td>OK</td>
</tr>
<tr>
<td>Student</td>
<td>tryPerson</td>
<td>name = &quot;Toni&quot;;</td>
<td>OK</td>
</tr>
<tr>
<td>Student</td>
<td>tryPerson</td>
<td>other.name = &quot;Suzanne&quot;;</td>
<td>OK</td>
</tr>
<tr>
<td>Student</td>
<td>tryStudent</td>
<td>other.name = &quot;Craig&quot;;</td>
<td>OK</td>
</tr>
</tbody>
</table>

END OF TEST