

Massachusetts Institute of Technology
6.170 Laboratory in Software Engineering

Spring 2007

Quiz 2

Wednesday, April 11, 2007

Name: _____

Athena username: _____

Section (circle one):

1: Lucy Mendel

2: Vikki Chou

3: David Glasser

4: Kah Seng Tay

5: Omair Malik

6: Clayton Sims

This quiz is closed book, closed notes. You have 50 minutes to complete it. It contains 27 questions in 14 pages (including this one), totaling 100 points. The last three pages contain duplicate material from multi-page questions. You may tear-off those pages for your reference. Before you start, please check your copy to make sure it is complete. Turn in pages 1-15, together, when you are finished. Separately turn-in pages 12-14.

Write your initials and section number on the top of ALL pages.

Please write neatly; we cannot give credit for what we cannot read.

Good luck!

Question(s)	Grading Scheme	Total	Max												
1 to 8	[] X 2 =	[]	of 16												
9 to 14	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px;">9</td> <td style="width: 20px;">10</td> <td style="width: 20px;">11</td> <td style="width: 20px;">12</td> <td style="width: 20px;">13</td> <td style="width: 20px;">14</td> </tr> <tr> <td>[]</td> <td>[]</td> <td>[]</td> <td>[]</td> <td>[]</td> <td>[]</td> </tr> </table> =	9	10	11	12	13	14	[]	[]	[]	[]	[]	[]	[]	of 30
9	10	11	12	13	14										
[]	[]	[]	[]	[]	[]										
15 to 18	[] X 2 =	[]	of 8												
19 to 23	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px;">19</td> <td style="width: 20px;">20</td> <td style="width: 20px;">21</td> <td style="width: 20px;">22</td> <td style="width: 20px;">23</td> </tr> <tr> <td>[]</td> <td>[]</td> <td>[]</td> <td>[]</td> <td>[]</td> </tr> </table> =	19	20	21	22	23	[]	[]	[]	[]	[]	[]	of 20		
19	20	21	22	23											
[]	[]	[]	[]	[]											
24 to 25	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px;">24</td> <td style="width: 20px;">25</td> </tr> <tr> <td>[]</td> <td>[]</td> </tr> </table> =	24	25	[]	[]	[]	of 10								
24	25														
[]	[]														
26 to 27	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px;">26</td> <td style="width: 20px;">27</td> </tr> <tr> <td>[]</td> <td>[]</td> </tr> </table> =	26	27	[]	[]	[]	of 16								
26	27														
[]	[]														
		[]	of 100												

Question 1. (circle True or False)

In general, designing a system such that its Module Dependency Diagram has more edges is good.

True / False

[F. It is better for things to be decoupled.]

Question 2. (circle True or False)

The observer pattern implements PUSH (as opposed to PULL) functionality.

True / False

[T. From lecture 16 slide 23.]
operations, what data is needed, and other factors.]

Question 3. (circle True or False)

Adding a precondition weakens the spec, whereas specifying that an exception is thrown strengthens the spec.

True / False

True

Question 4. (circle True or False)

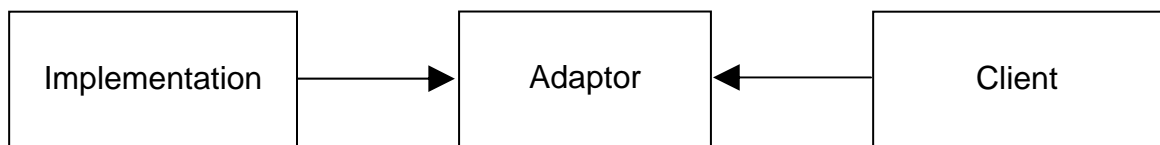
Implementing an Adaptor pattern through subclassing allows us to have multiple wrappers.

True / False

(False: Delegation allows us that)

Question 5. (circle True or False)

The following is a valid MDD for an adapter design pattern:



True / False

(False: Adaptor should subclass Implementation, not the other way round)

Question 6. (circle True or False)

If a system does not allow the addition of more types, a procedural pattern is preferred to an interpreter pattern.

True / False

(True)

Question 7. (circle True or False)

Assertions should never be enabled in production code.

True / False

(False)

Question 8. (circle True or False)

"Fail-fast" behavior helps to localize bugs.

True / False

(True)

Question 9. (Circle all that apply)

Which of the following are appropriate uses of assert statements:

- (A)

```
int oldSize = myList.size();
myList.add(element);
assert myList.size() == oldSize+1;
```
- (B)

```
assert myList.add(element);
```
- (C)

```
/** @requires element != null */
public void add(E element) {
assert element != null;
}
```
- (D)

```
public void add(E element) {
    ...
assert repOK();
return true;
}
```

C and D are the correct answers. A is false because it clutters the code.
B is false because it side-effects.

Question 10. (Circle all that apply)

Alyssa decides to employ a top-down approach for the project. Which of her following reasons are true for a top-down process:

- (A) A top-down process is more time consuming because of the unit tests.
- (B) Alyssa likes top-down because she can present a demo of the project to the management faster than using a bottom-up process.
- (C) In a top-down design, if an error is detected it's always because a lower-level module is not meeting its specifications (because the higher-level ones are already been tested).
- (D) A top-down process makes it possible to detect performance problems faster
- (E) A top-down process makes it easier to fix a global conceptual problem

B, E

Question 11. (Circle the most appropriate)

A program took 160 seconds to execute on a single processor but only 64 seconds on a 4 core multicore. What is the best estimate for the execution time on a 64 core machine?

- (A) 2 seconds
- (B) 16 seconds
- (C) 22 seconds
- (D) 34 seconds
- (E) 48 seconds

Ans: D $a + b = 160$, $a + b/4 = 64 \rightarrow a = 32$; $b = 128$; $a + b/64 = 32 + 128/64 = 34$

You have accepted a consulting job at Google to help them debug their online library. They ask you to prove that the rep invariant holds for the following code:

```
/**
 * Book represents a physical book containing multiple bookmarks, one for each reader.
 */
public class Book {

    // RI: pages > 0 AND
    // all elements of bookmarks.values() are > 0 and <= pages

    private final Number pages;
    private final Map<Person, Number> bookmarks;

    public Book(Number pages) {
        this.pages = pages;
        bookmarks = new HashMap<Person, Number>();
    }

    public void readPage(Person p) {
        Number n = bookmarks.get(p);
        if (n == null) {
            bookmarks.put(p, new Number(1));
        } else if (!n.equals(pages)) {
            bookmarks.put(p, n.incr())
        }
    }

    public void resetBookmark(Person p) {
        bookmarks.put(p, new Number(1));
    }

    public Book copy() {
        Book other = new Book(pages);
        for (Person p : bookmarks.keySet()) {
            for (int i = 0; i < bookmarks.get(p); i++) {
                other.readPage(p);
            }
        }
        return other;
    }
}
```

Correction: Number is not java.lang.Number. Assume it is an immutable class that can represent arbitrary integers and has methods such as incr.

Question 12. (Circle all that apply)

Using the approach taught in lecture, which of the following methods/constructors would be treated as base cases in an inductive proof of the rep invariant:

- (A) Book
- (B) readPage
- (C) resetBookmark
- (D) Number
- (E) copy

The answer is A, since only the constructor is the base case (copy need only show that it doesn't mutate the book)

Question 13. (Circle all that apply)

Using the approach taught in lecture, for which of the following methods/constructors would the inductive step need to hold in an inductive proof of the rep invariant:

- (A) Book
- (B) readPage
- (C) resetBookmark
- (D) copy
- (E) KeySet

B, C, D -- all mutators/observers/producers could mutate the object, so the inductive step must be proven to hold for all.

Question 14. (Circle all that apply)

Can the rep invariant be proved to hold?

- (A) No, due to copy
- (B) No, due to resetBookmark
- (C) No, due to readPage
- (D) No, due to Book
- (E) Yes

D – does not check for non-negative numbers

Suppose we modify the RatNum class from PS1 so that it is not final, and we make a new class ImaginaryRatNum that extends RatNum.

```
public class ImaginaryRatNum extends RatNum {

    private boolean isImaginary = false;

    public ImaginaryRatNum(boolean isImaginary, int n) {
        super(n);
        this.isImaginary = isImaginary;
    }

    public boolean equals(RatNum n) {
        // if equal zero
        if ( this.numer == 0 && n.numer == 0) {
            return true;
        } else {
            if (isImaginary) {
                return false;
            } else {
                return super.equals(n);
            }
        }
    }
}
```

Correction: This method does not override Object equal. Thus, you can assume that there is a method

```
public boolean equals(Object o) {
    if (!(o instanceof RatNum))
        return false;
    else
        return equals((RatNum) o);
}
```

For the following equality contracts that each Java object should obey, which of those ImaginaryRatNum class satisfy or fails to satisfy.

Question 15. (circle Satisfy or Fails to satisfy)

Reflexivity Satisfy / Fails to satisfy

reflexivity - not satisfied because an ImaginaryRatNum 0 will not equal itself

Question 16. (circle Satisfy or Fails to satisfy)

Symmetry Satisfy / Fails to satisfy

symmetry - not satisfied because an ImaginaryRatNum 0 may equal RatNum 0 but not the other way round.

Question 17. (circle Satisfy or Fails to satisfy)

Transitivity Satisfy / Fails to satisfy

transitivity - satisfied

Question 18. (circle Satisfy or Fails to satisfy)

Consistency Satisfy / Fails to satisfy

consistency - satisfied

The American Idol TV program has changed its voting scheme. Instead of voting for one candidate, the callers are asked to indicate “candidate A is better than candidate B”. Initially each candidate is given 10000 points and for each vote “A is better than B”, 1% of points is deducted from candidate B’s total and added to candidate A’s total. You, as the wiz programmer for American Idol, have to implement the code that does the vote tallying. Since millions of people dial-in, to handle the peak load your code has to run on hundreds of processors simultaneously.

```
public class contestant {
    // RI: points >= 0
    int points;
    public final int myid;        // unique id of each contestant
    // Assume there is a constructor that initiates all the fields correctly.
    void post(int i)      { points += i; }
    int get()            { return points; }
}

public class AmericanIdol extends thread {
    // RI: SUMfor_all_contestants points = 10000 * number_of_contestants
    static AmericanIdol AI; // the game object
    contestant con[ ];
    // Assume there are methods that initiates all the fields correctly and start the threads
    public void vote(contestant from, contestant to) {
        int mov;
        <BODY>
    }
}
```

For the next five questions, consider the following code for the <BODY> of the vote method.

A <pre>synchronized(AI) { mov = from.get()/100; from.post(-mov); to.post(mov); }</pre>	D <pre>synchronized(from) { mov = from.get()/100; from.post(-mov); to.post(mov); }</pre>
B <pre>synchronized(from) { mov = from.get()/100; } synchronized(from) { from.post(-mov); } synchronized(to) { to.post(mov); }</pre>	E <pre>synchronized(from) { synchronized(to) { mov = from.get()/100; from.post(-mov); to.post(mov); } }</pre>
C <pre>synchronized((from.myid > to.myid)?from:to) { synchronized((from.myid > to.myid)?to:from) { mov = from.get()/100; from.post(-mov); to.post(mov); } }</pre>	

Question 19. (Circle all that apply)

Many voters will not be able to vote because the system will freeze (i.e. deadlock) or will be slow to respond (i.e. will not scale).

What <BODY>(s) will lead to this problem?

- (A) (B) (C) (D) (E)

A and E: A because it locks the entire game, making it sequential. E because it can deadlock

Question 20. (Circle all that apply)

The class AmericanIdol's RI: $\text{SUM}_{\text{for_all_contestants}} \text{points} = 10000 * \text{number_of_contestants}$ will be not be maintained

What <BODY>(s) will lead to this problem?

- (A) (B) (C) (D) (E)

D because it has a race condition (Not B, because even if the number of points of a candidate is negative the exact amount added to a candidate is subtracted from the other.)

Question 21. (Circle all that apply)

In a heavily contested election (a lot of callers) the vote can be biased against some contestants by the program. For example, if votes for a contestant are consistently delayed more than the rest, the program may be biased against him or her.

What <BODY>(s) will lead to this problem?

- (A) (B) (C) (D) (E)

C because the contestants with a higher myid are in critical sections longer than ones with lower myid.

Question 22. (Circle all that apply)

Results will be unreliable because of a race condition

What <BODY>(s) will lead to this problem?

- (A) (B) (C) (D) (E)

D because the 'to' object is modified outside a critical section, thus multiple invocations of vote can simultaneously modify it.

Question 23. (Circle all that apply)

The class contestant's RI: $\text{points} \geq 0$ will not be maintained.

What <BODY>(s) will lead to this problem?

- (A) (B) (C) (D) (E)

B because reading the points and updating is not atomic. If more than 100 separate calls to vote read the points and then proceed to update (subtract 1%) the final value will be negative. (not D because the update that reduces the points are under a critical section. Only updates that can increase the points will be interleaved. Thus, any increment between read and decrement cannot make the points negative)

```
long power(long x, long y) {
    // P
    long e = y;

    List<Long> binary = new ArrayList<Long>();
    while (e>0) {
        binary.add(e%2);
        e = (long) (e/2);
    }
    long r = 1;
    Long[] binaryArray = binary.toArray(new Long[ ] { });
    for (int i = binaryArray.length-1; i >= 0; i--) {
        long d = binaryArray[i].longValue();
        // S1
        if (d % 2 == 1) {
            r = r*r * x;
        } else {
            r = r*r
        }
        // S2
    }
    // Q
    return r;
}
```

Question 24. (circle all that apply)

- (A) Both while and for loops terminate
- (B) Only the while loop is guaranteed to terminate
- (C) The while loop terminates only when $y > 0$
- (D) While loop never terminates
- (E) Insufficient information to determine if any of the loops terminate

Ans: A

Question 25. (circle the most appropriate)

Which of the following is the decrementing function?

- (A) x
- (B) y
- (C) r
- (D) e
- (E) $r*r$

Correction: for the while loop

Ans: D

Question 26.

Suppose at one iteration of the loop, we have

S2 : $r = 64$,
 $e = \text{some known value } C$

Correction: $d = \text{some known value } C$

What is the weakest predicate for S1 in that same iteration of the loop?

$d \text{ even}$ $r^2 = 64$ $d = C$

$d \text{ odd}$ $r^2 \times x = 64$ $d = C$

Question 27.

Suppose we have

Q: $r > 0$.

What is the weakest predicate for P?

$(y \text{ even and } x \neq 0) \text{ or } x > 0 \text{ or } y \leq 0$

Code for the questions 12, 13 and 14

```
/**
 * Book represents a physical book containing multiple bookmarks, one for each reader.
 */
public class Book {

    // RI: pages > 0 AND
    // all elements of bookmarks.values() are > 0 and <= pages

    private final Number pages;
    private final Map<Person, Number> bookmarks;

    public Book(Number pages) {
        this.pages = pages;
        bookmarks = new HashMap<Person, Number>();
    }

    public void readPage(Person p) {
        Number n = bookmarks.get(p);
        if (n == null) {
            bookmarks.put(p, new Number(1));
        } else if (!n.equals(pages)) {
            bookmarks.put(p, n.incr())
        }
    }

    public void resetBookmark(Person p) {
        bookmarks.put(p, new Number(1));
    }

    public Book copy() {
        Book other = new Book(pages);
        for (Person p : bookmarks.keySet()) {
            for (int i = 0; i < bookmarks.get(p); i++) {
                other.readPage(p);
            }
        }
        return other;
    }
}
```

Code for the questions 19 to 23

```

public class contestant {
    // RI: points >= 0

    int points;
    public final int myid;        // unique id of each contestant

    // Assume there is a constructor that initiates all the fields correctly

    void post(int i)    { points += i; }

    int get()          { return points; }
}

public class AmericanIdol extends thread {
    // RI: SUMfor_all_contestants points = 10000 * number_of_contestants

    static AmericanIdol AI; // the game object
    contestant con[ ];

    // Assume there are methods that initiates all the fields correctly and start the threads

    public void vote(contestant from, contestant to) {
        int mov;
        <BODY>
    }
}

```

A	<pre> synchronized(AI) { mov = from.get()/100; from.post(-mov); to.post(mov); } </pre>	B	<pre> synchronized(from) { mov = from.get()/100; } synchronized(from) { from.post(-mov); } synchronized(to) { to.post(mov); } </pre>
C	<pre> synchronized((from.myid > to.myid)?from:to) { synchronized((from.myid > to.myid)?to:from) { mov = from.get()/100; from.post(-mov); to.post(mov); } } </pre>		
D	<pre> synchronized(from) { mov = from.get()/100; from.post(-mov); to.post(mov); } </pre>	E	<pre> synchronized(from) { synchronized(to) { mov = from.get()/100; from.post(-mov); to.post(mov); } } </pre>

Code for the questions 24 to 27

```
long power(long x, long y) {
    // P
    long e = y;

    List<Long> binary = new ArrayList<Long>();
    while (e>0) {
        binary.add(e%2);
        e = (long) (e/2);
    }
    long r = 1;
    Long[] binaryArray = binary.toArray(new Long[] { });
    for (int i = binaryArray.length-1; i >= 0; i--) {
        long d = binaryArray[i].longValue();
        // S1
        if (d % 2 == 1) {
            r = r*r * x;
        } else {
            r = r*r
        }
        // S2
    }
    // Q
    return r;
}
```