

6.170: Quiz 2

- Don't open this booklet until you are directed to do so.
- Write your name on the front page.
- Circle your TA's name and your section number.
- This quiz is 50 minutes long. It contains **7** questions and has **14** pages. Please check your copy to make sure it is complete before you start.
- Good luck!

Problem	Grade	Points
1		12
2		12
3		20
4		12
5		14
6		20
7		10
Total		100

Circle your TA and Section #:

Section 1: Matt Section 2: Roshan
Section 3: Kalpak Section 4: Brandy
Section 5: Allen Section 6: Todd
Section 7: Kurt Section 8: Godfrey
Section 9: Jon

Name: _____

1 Problem Object Models (12 points)

1.1 We wish to create a meta-model for our object modelling notation as described in lecture. We will focus on representing the subset relationship. Draw a (meta) object model for object models that can represent and distinguish between

- Subsets A and B of a set C , where A and B can have common elements.
- Disjoint subsets A and B of a set C .
- Disjoint subsets A and B that exhaustively enumerate a set C .

1.2 We wish to build a hierarchical database. The database is made up of many *Entry* objects. There are only two kinds of entries, *Composite* and *Leaf*. Composite entries contain at least one entry. Leaf entries do not contain any entries. Each entry maps to *data*, and this mapping is mutable. However, data has to be destroyed and recreated if it is to be associated with another entry. The mapping from a composite entry to its constituent entries is immutable.

2 Subtyping (12 points)

Examine each of the following classes, then answer the questions below.

```
class ProjectPartner {
    // requires: hours <= 25
    // effects:  complete hours amount of work
    // throws:  SlippageException if the work can't be completed ontime
    void work(int hours) { ... }
}

class SamSlacker {
    // requires: hours <= 25
    // effects:  complete hours / 2 amount of work
    // throws:  SlippageException if the work can't be completed ontime
    void work(int hours) { ... }
}

class CornieConscientious {
    // requires: hours <= 25
    // effects:  complete hours amount of work
    // throws:  SickException when falling sick
    void work(int hours) { ... }
}

class ArnieClever {
    // requires: hours <= 25
    // effects:  complete at least hours amount of work
    // throws:  none
    void work(int hours) { ... }
}

class JoeWhatever {
    int mood; // modifiable level of mood
    // requires: none
    // effects:  complete 20 <= hours <= 30 amount of work depending on mood
    // throws:  none
    void work(int hours) { ... }
}
```

2.1 For each of the following classes, determine whether the class can be a true subtype of ProjectPartner and give a **one sentence** explanation to support your answer.

a) SamSlacker

b) CornieConscientious

c) ArnieClever

d) JoeWhatever

2.2 List all of the aforementioned classes that can be a Java subclass of ProjectPartner. No explanation is necessary.

3 Dynamic Reasoning (20 points)

This is a difficult question!

Recall from lecture that:

```
wp(S, Q) is the weakest predicate such that
  wp(S, Q) {S} Q holds
where
  S is the code being executed, and
  Q is the post-assertion.
```

3.1 Reduce the following weakest precondition formula to its simplest form:

```
wp({if (x < y)
    min = x;
    else
    min = y;
  y = 0; },
  y >= min)
```

3.2 Provide a loop invariant sufficient to prove the partial correctness of the following program fragment designed to find out whether or not `isPrime(x)`, where `isPrime(x)` is true if and only if there does not exist an integer, `j`, such that $1 < j < x$ and `remainder(x, j) = 0`. Note that `remainder(x, j)` of two integers, `x` and `j`, will return the integer remainder if x / j ; in other words, $x \% j$.

```
// assert y = 2 && foo = true && x > 0
while (y < x) {
    if (remainder(x, y) == 0)
        foo = false;
    y = y + 1;
}
// assert foo = isPrime(x)
```

3.3 Provide a decrementing function that can be used to prove the total correctness of the program fragment in part 3.2.

4 Equality (12 points)

Look at the two classes Point and LineSegment below:

```
final public class Point {

    private int x;
    private int y;

    public Point(int x_coord, int y_coord) {
        x = x_coord;
        y = y_coord;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}

public class LineSegment {

    private Point start;
    private Point end;

    public LineSegment(Point a, Point b) {
        start = a;
        end = b;
    }

    public void setStart(Point a) {
        start = a;
    }

    public Point setEnd(Point b) {
        end = b;
    }

    ...observer methods not shown...
}
```

4.1 Write an “equals” method and “similar” method for class Point, according to Liskov’s approach towards equality.

4.2 Write an “equals” method and “similar” method for class LineSegment, according to Liskov’s approach towards equality.

5 Inheritance (14 pts)

5.1 Which of the following is the “general rule of thumb” that should be used to determine when one *can* use inheritance? (Circle one)

1. Use inheritance whenever you want to reuse code that is implemented in another class.
2. Use inheritance if the name of a class is similar to the name of another class.
3. Use inheritance if two classes satisfy the mutual decoupling rule.
4. Use inheritance if two classes satisfy the substitution rule for subtyping.

5.2 Examine the following piece of code. Determine if the class `TimTheBeaver` is a *true* subtype of the class `Jedi`. In one or two sentences, give an explanation to justify your answer.

```
class Jedi {
    // effects: prints "May the force be with you." to stdout
    public void a() {
        b();
    }

    // effects: prints "May the force be with you." to stdout
    public void b() {
        System.out.println("May the force be with you.");
    }
}

class TimTheBeaver extends Jedi {
    // effects: prints "May the force be with you." to stdout
    public void b() {
        super.a();
    }

    public static void main(String[] args) {
        Jedi apprentice = new TimTheBeaver();
        apprentice.b();
    }
}
```

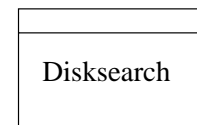
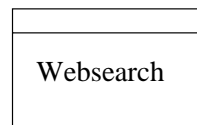
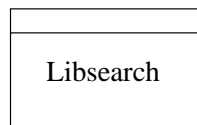
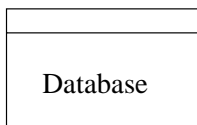
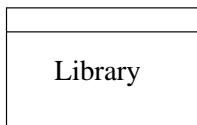
5.3 In one to two sentences, explain what happens when the `main()` in `Jedi` gets run.

6 Design Patterns (20 points)

In this problem, you will be asked to think about the design of two small pieces of a system. We will ask you to draw several MDD's of elements of the system. You can collect together functionality of several classes in a single module if you wish. We will be providing you with skeletal MDD's for each piece, which you will be filling in.

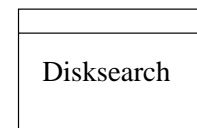
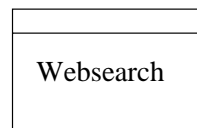
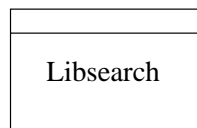
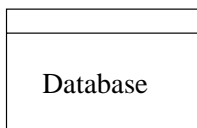
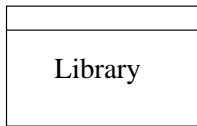
The system in question is a library management system. This system manages databases of the books in the library, and the card-carrying customers of the library.

6.1 The library system contains a database subsystem. The database uses several search algorithms such as `libsearch`, `websearch`, and `disksearch`. Fill in the dependences in the MDD below corresponding to the database part of the library system.



6.2 The user of the library system wishes to specify the search facility after logging on, before he or she accesses the database(s). Using a design pattern, develop a more modular architecture for the library system that minimizes the coupling between the modules.

Fill in the MDD below to model your modular architecture. Feel free to add more modules or interfaces to the MDD as necessary. Include the name of the design pattern you are using.



7 Project Management (10 points)

True/false questions.

- 7.1 **True / False** For an incorrectly implemented method, zero documentation is better than falsified documentation.
- 7.2 **True / False** Reasonable documentation, but with no versioning and notion of authorship, can help problem resolution between two authors working on the same code.
- 7.3 **True / False** Keeping legacy code is always bad because cutting-edge software can easily replace it and reduce maintenance cost.
- 7.4 **True / False** NASA is in a race with other space agencies to be the first to send a human to Mars. For such a massive undertaking involving hundreds of developers, their management's decision to have a decentralized team is the right solution for building the software system.
- 7.5 **True / False** NASA management decides that the hacking model to build the software components will give best results, since all the team members are highly-acclaimed hackers from MIT.
- 7.6 **True / False** For a medium to large-sized project, the Prototyping/Incremental model is better than the hacking model when customers are different from the developers.
- 7.7 **True / False** The Prototyping phase of the Prototyping/Incremental model is like the hacking model.
- 7.8 **True / False** Even though a prototype in the Prototyping/Incremental model has a functional mapping with a module in the real system, we must throw the prototype away.
- 7.9 **True / False** The Prototyping/Incremental model of development reduces total expected post-deployment error correction costs.
- 7.10 **True / False** The 6.170 final project is a lot of fun.

The end!