Final Exam

Software Engineering

810:172

Fall 2009

Instructions

- The exam consists of this cover sheet, eleven (11) problems, and two (2) pages of code. Be sure that you have all of these items and that they are all legible.
- Read all questions and their instructions thoroughly before you begin. It is always worth your time to plan ahead!
- Write your answers in the spaces provided on the exam.
- Whenever I ask you to answer "... *briefly*...", a one- to three-sentence answer is long enough to express the each bullet item.
- Whenever a problem contains multiple parts, be sure that your response includes answers to each part.
- The exam is worth sixty (60) points total. The point value of each question is given in brackets immediately after the problem number. You may wish to use these point values as you budget your time across the exam period.
- Points will be awarded based on your explicit answers. Partial credit will be given where possible, so show all of your work.
- The exam lasts sixty (60) minutes. It is due at 4:00 PM.

Name:

1. [6 points] Answer *briefly* each of these questions about refactoring.

- What **is** refactoring?
- What role does refactoring play in the design of software?
- What is the relationship between refactoring and testing?

2. [6 points] In class, we began to refactor a `Customer` class for a video rental store system. The current version is given at the end of the exam.

- Describe two different refactorings that we could apply to improve this code by reducing duplication between the `statement()` and `htmlStatement()` methods.
- For each of the refactorings, identify at least one programming issue that we would face in trying to ensure that we don't break the program.

3. [6 points] Answer *briefly* these questions about the claim that the best programmers are up to 28 times more productive than the worst programmers.

- What factors lead to such variation between the best and worst programmers?
- Identify a tool or process we can use to help programmers be more productive.
- What potential problems does use of this tool or process create?

4. [4 points] Answer *briefly* each of these questions about test-first development.

- Identify two advantages that come with writing tests before code.
- What trade-off does a tester make between looking at the code *before* writing tests and *after* writing tests?

5. [6 points] Answer *briefly* each of these questions about the "Writing Maintainable Automated Acceptance Tests" article by Emery.

- What relationship between maintenance and testing does Emery address?
- Describe the difference between the acceptance test at the beginning of the paper and the acceptance test at the end.
- What tools are needed to implement the sort of tests that Emery describes?

6. [4 points] Answer *briefly* these questions about testing as an attempt to demonstrate a match between the specification for a system and the code that implements the system.

- Identify two clues a tester can take from the specification when writing tests.
- Identify two clues a tester can take from the code when writing tests.

7. [6 points] Answer *briefly* these questions about agile methodology, as it relates to Fowler's distinction between *predictive* versus *adaptive* approaches to software development processes.

- Describe at least two ways in which agile methodologies are adaptive rather than predictive.
- What risk does a development team incur when it chooses to be adaptive rather than predictive?
- In what ways does extreme programming try to mitigate this risk?

8. [6 points] Based on your experience with your team project, answer *briefly* these questions about the software life cycle.

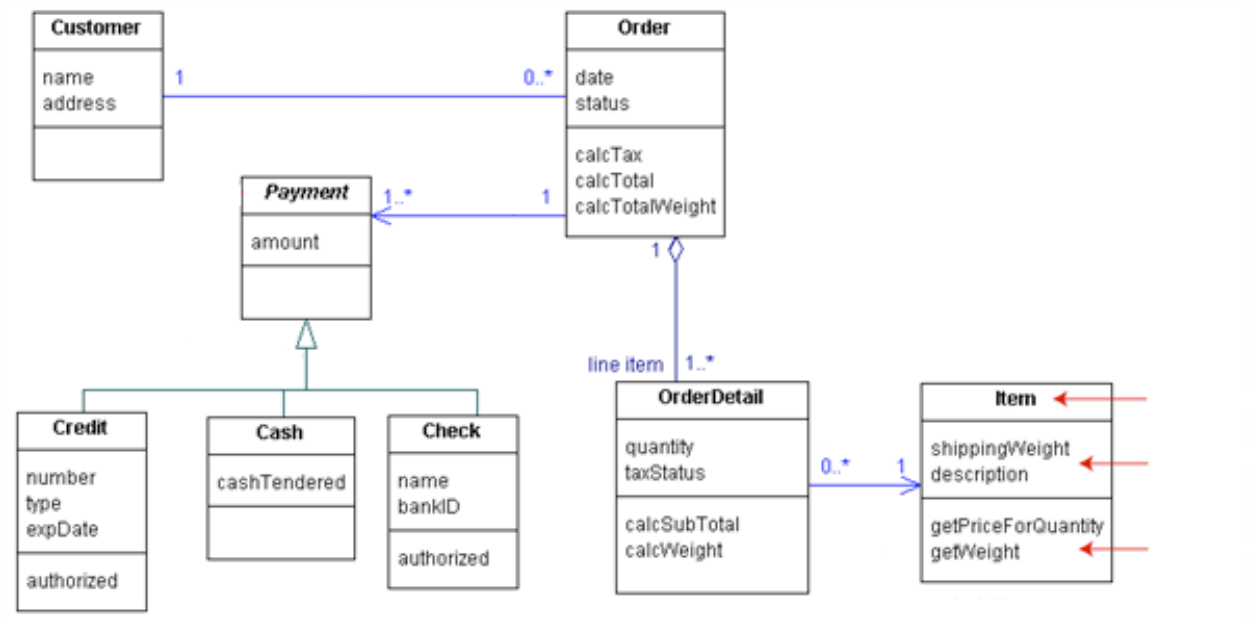
- Which stage was most difficult? Why? What did your team do in response?
- Which stage was most straightforward? Why?

9. [6 points] In light of your experience building a medium-sized program with a team, discuss *briefly* the key ideas in the "No Silver Bullet" article by Brooks. Be sure to discuss at least two of Brooks's ideas.

10. [4 points] Answer *briefly* these questions about specification.

- What is a *functional* specification?
- What is a *technical* specification?

11. [6 points] Identify each of the labeled components in this UML class diagram.



A _____ F _____

B _____ G _____

C _____ H _____

D _____ I _____

E _____ J _____

.

```
import java.util.Enumeration;
import java.util.Vector;

public class Customer
{
    private String name;
    private Vector rentals;

    public Customer( String name )
    {
        this.name    = name;
        this.rentals = new Vector();
    }

    public void addRental( Rental arg )
    {
        rentals.addElement( arg );
    }

    public String getName()
    {
        return name;
    }

    public String statement()
    {
        double totalCharge          = 0;
        int    frequentRenterPoints = 0;
        String result  = "Rental Record for " + getName() + "\n";

        Enumeration entries = rentals.elements();
        while ( entries.hasMoreElements() )
        {
            Rental each = (Rental) entries.nextElement();
            double rentalCharge = each.charge();
            frequentRenterPoints += each.frequentRenterPoints();

            // show figures for this rental
            result += "\t" + each.getMovie().getTitle() +
                      "\t" + String.valueOf(rentalCharge) + "\n";
            totalCharge += rentalCharge;
        }

        result += "Amount owed is " + String.valueOf(totalCharge) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) +
                  " frequent renter points";

        return result;
    }
```

```
    public String htmlStatement()
    {
        double totalCharge          = 0;
        int    frequentRenterPoints = 0;
        String result  = "<HTML>\n<TITLE>Rental Record</TITLE>\n" +
                         "<BODY>\n" +
                         "<H1>Rental Record for <EM>" + getName() +
                         "</EM></H1>\n<P>";

        Enumeration entries = rentals.elements();
        while ( entries.hasMoreElements() )
        {
            Rental each = (Rental) entries.nextElement();
            double rentalCharge = each.charge();
            frequentRenterPoints += each.frequentRenterPoints();

            // show figures for this rental
            result += each.getMovie().getTitle() + ":" +
                      String.valueOf(rentalCharge) + "<BR/>\n";
            totalCharge += rentalCharge;
        }

        result += "</P>\n<P> You owe <EM>" + String.valueOf(totalCharge) +
                  "</EM></P>\n<P>On this rental, you earned <EM>" +
                  String.valueOf(frequentRenterPoints) +
                  "</EM> frequent renter points.</P>\n</BODY>\n</HTML>";

        return result;
    }
}
.
```