COMP/EECE 7012

Exam 2

Spring 2014

Name ("Last, First"): Solutions

Rules:

- No potty breaks.
- Turn off cell phones/devices.
- Closed book, closed note, closed neighbor.

- WEIRD! Do not write on the backs of pages. If you need more pages, ask me for some.

Reminders:

- Verify that you have all pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

Additional Items:

- For questions that involve writing code:
  - You may omit `import` statements.
  - You may omit exception-handling code.

1. [2pts] What often-false assumption does the waterfall model made about requirements specifications?

   a. Specifications are predictable

   b. Specifications are stable

   c. Specifications have low change rates

   [✓ d.] All of the above

   e. None of the above

2. [2pts] True or false? It is better to discover defects later in the development process. That way, you can have more of the system finished before you worry about fixing things.

   a. True

   [✓ b.] False

3. [2pts] An empirical process model iterates between…

   [✓ a.] … feedback and adaptation

   b. … design and implementation

   c. … requirements gathering and design

   d. … user studies and testing

   e. None of the above

4. [2pts] True or false? In iterative software development, it is recommended that iterations be 3 to 6 months in length.

   a. True

   [✓ b.] False

5. [6pts] Think of the eCourseware system that we've used in class. Reverse engineer one user story that records a requirement for the system. You must apply the description template described in class, and your US must have the other attributes of good user stories, which we discussed in class.

Many possible answers...

Here's the template:

"As a <who>, I want <what> <why>."

Here are some other good attributes:

- Describe one thing
- Use customers language
- Not be long essay
- Not use technical terms

6. [2pts] Which of the following is a *non-functional requirement*?

   a. The system enables users to place lunch orders.

   [✓ b.] The system always responds to user clicks in less than one tenth of a second.

   c. The system displays a list of hotel vacancies.

   d. The system notifies the user when a new order arrives.

   e. None of the above

7. [7pts] Describe the process of iteration planning that we used in this course by writing 7 sentences. Create each sentence by filling in 3 blanks with the following words/phrases. Fill in *all* blanks.
   a. Blank #1: developer, customer
   b. Blank #2: estimates, selects (for iteration), assigns (to developer), creates, prioritizes
   c. Blank #3: tasks, user stories

developer (or customer)	creates	user stories.
developer	estimates	user stories.
customer	prioritizes	user stories.
developer	selects	user stories.
developer	creates	tasks.
developer	estimates	tasks.
developer	assigns	tasks.

8. [2pts] According to the INVEST mnemonic, which of the following is a good quality of a user story?

   a. USs are independent of one another

   b. USs have value to the customer

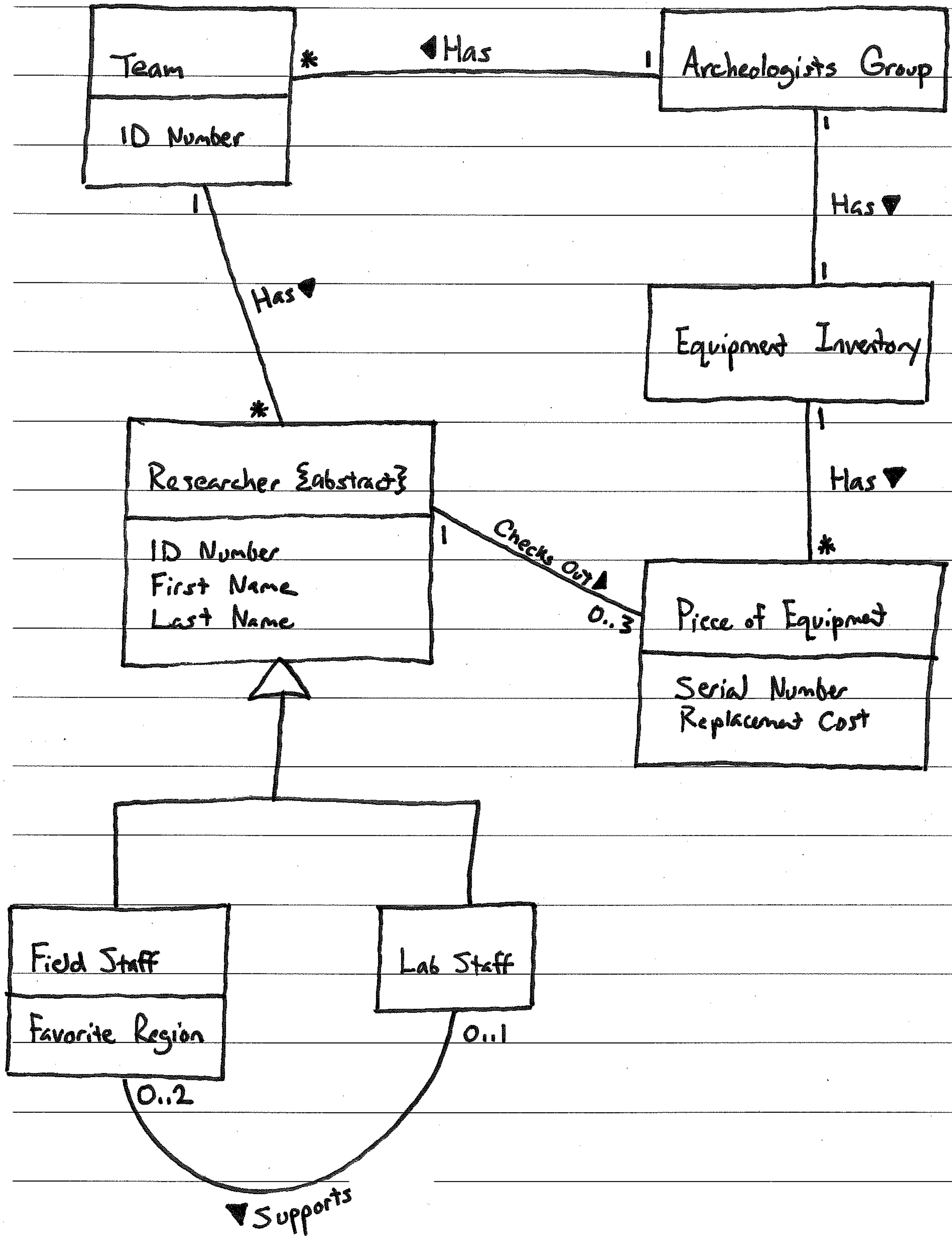
   c. USs are generally small

   [✓ d.] All of the above

   e. None of the above

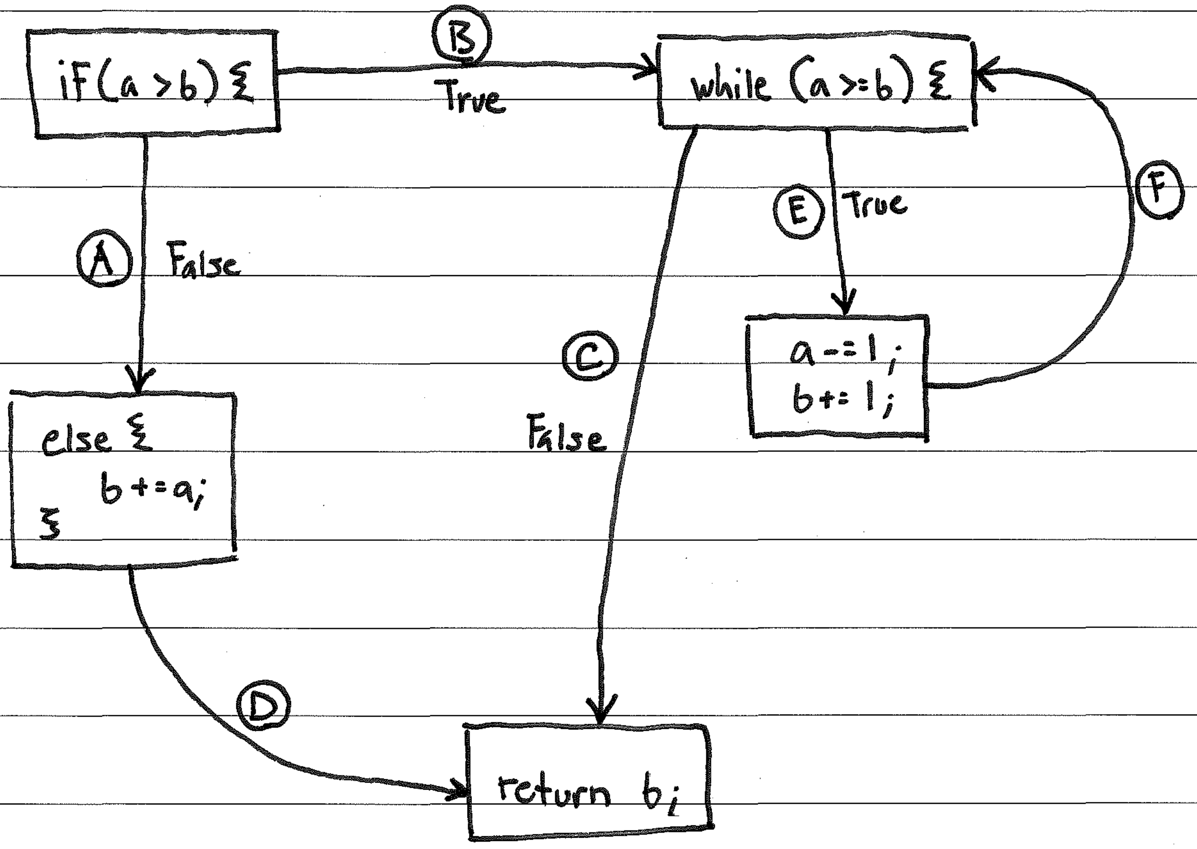
9. [10pts] On the next page, create a domain model (using class diagram notation) based on the following description.

   You have been asked to build a management system for a group of archeologists. The group is comprised of multiple teams of researchers. Each team has a letter ID (e.g., team A, team B). Each researcher belongs to one of the teams, and has an ID number, a first name, and a last name. There are two types of researchers: field and lab staff. Each field staff member has a favorite region (string). Each lab researcher supports up to 2 field researchers. Some researchers may not be supported by a lab researcher. The company also manages an inventory of equipment. Researchers of any type may check out up to 3 pieces of equipment. Each piece of equipment has a serial number and replacement cost.


Team
ID Number
*
◀Has
1
Archeologists Group
1
Has▼
1
Equipment Inventory
1
Has▼
*
1
Has▼
*
Researcher {abstract}
ID Number
First Name
Last Name
1
Checks Out▲
0..3
Piece of Equipment
Serial Number
Replacement Cost
Field Staff
Favorite Region
Lab Staff
0..1
0..2
▼Supports

10. [8pts] Draw a control flow diagram for this function. Label each edge with an uppercase letter.

```
int funWithNumbers(int a, int b) {
    if (a > b) {
        while (a >= b) {
            a -= 1;
            b += 1;
        }
    } else {
        b += a;
    }
    return b;
}
```



11. [5pts] Fill in the table below with a test suite that provides *path coverage* of the code from the previous question. Cover no more than 2 iterations of the loop. In the covers column, list the relevant labeled items in your CFG that each test case covers. If there is some part of the coverage that is impossible to cover, then list it in the covers column, and put "N/A" in the associated x and y cells. Some cells in the table may be left blank.

Input		**Covers**
x	y	
1	2	AD
N/A	N/A	BC
1	0	BEFC
4	2	BEFEFC

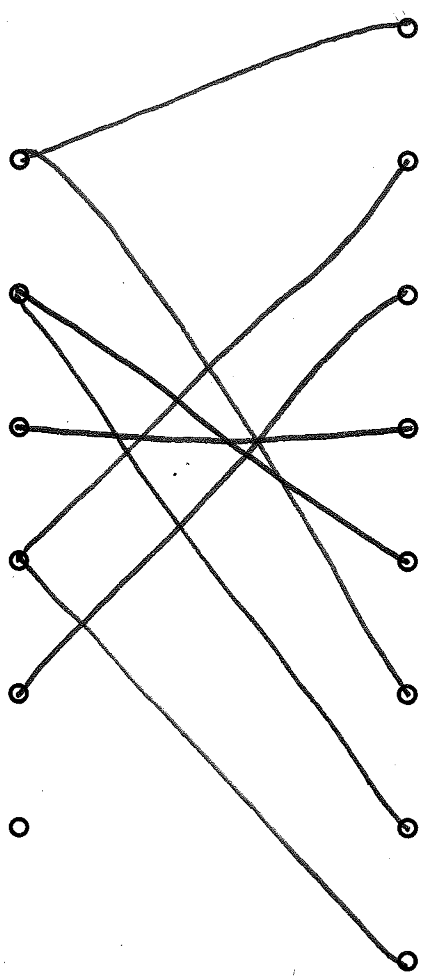
12. [4pts] Give two reasons why it's bad for developers to system test their own code.

Many possible answers...

Here ~~[illegible]~~ two good ones:

- Developer can't see system like a user does (he/she knows too much about its inner workings)
- Because developer made it, he/she may have disincentive to find problems

13. [8pts] Match the design pattern to the situation to which you should apply it.



Builder

Memento

Façade

Adapter

Interpreter

Observer

Your application needs to generate HTML files (from scratch).

Your program must support switching among several different email libraries, but each one has a slightly different interface.

You want to let users create and run macros inside your application.

Sending an SMS message requires lots of big, ugly code, involving connection, message, and other objects.

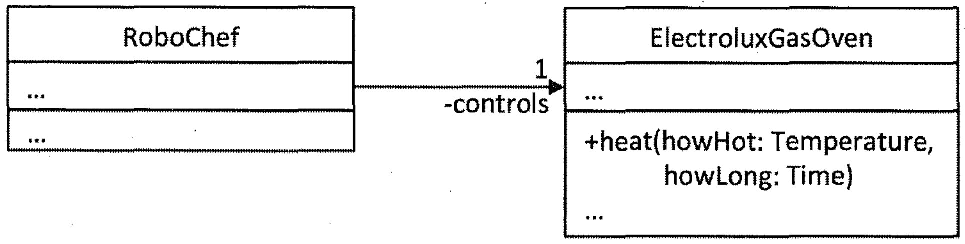
You want your application to save its state so that if it crashes, then it can auto-recover.

Your program has to create and configure some big, ugly record objects before inserting them into a database.

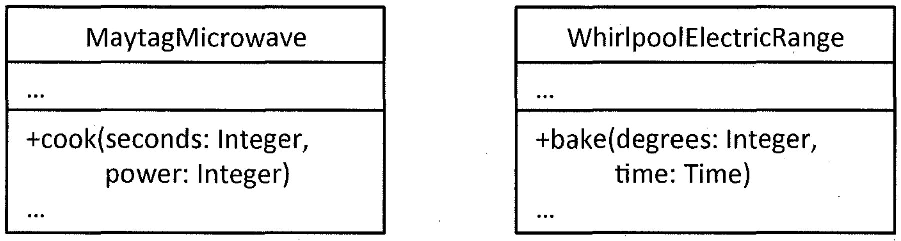
Your program has to support replication. You need a way for the program to save its state so the program can be copied to other servers.

Your company already implemented a component that almost implements the interface that you need, but not quite.

14. [10pts] Imagine that you are the creator of an "intelligent" kitchen system, RoboChef, that can actually control different kitchen appliances (e.g., ovens, choppers) to prepare food. Initially, you implemented RoboChef to use only Electrolux gas ovens. Here is an excerpt of your current software design:

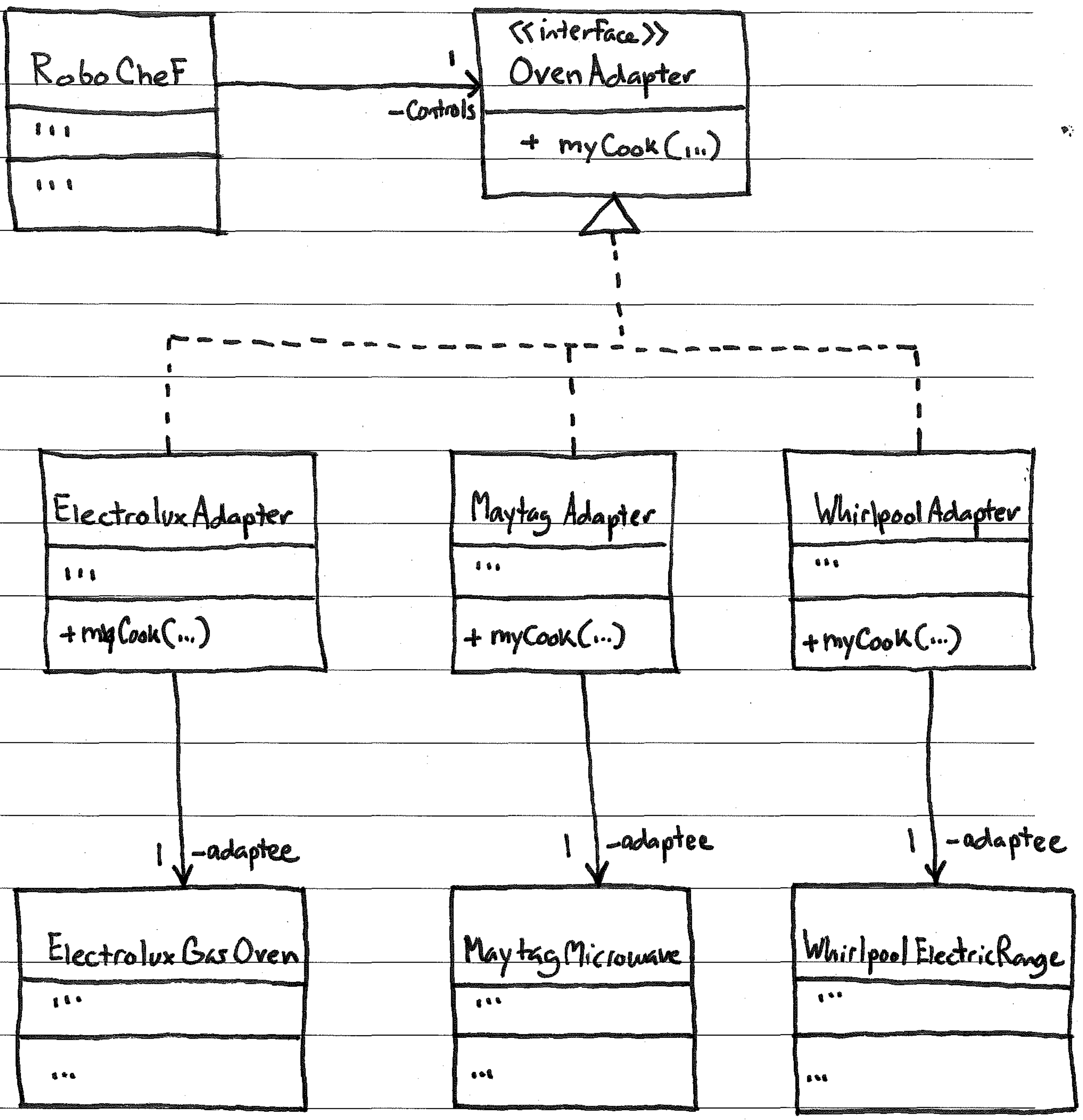


Note that the Electrolux Company provided the software interface for controlling the gas oven (ElectroluxGasOven), and you created the intelligent decision-making part (RoboChef). As your next step, you would like your system to support different types of ovens other than Electolux gas ones. For example, Maytag and Whirlpool each provide their own software interfaces for their ovens:


MaytagMicrowave
...
+cook(seconds: Integer, power: Integer)
...
WhirlpoolElectricRange
...
+bake(degrees: Integer, time: Time)
...

Update your current software design to allow easy switching between oven-control systems. Your design must apply a combination of the following patterns discussed in class, which are not entirely orthogonal: the **Adapter Pattern**, the **Indirection Pattern**, the **Protected Variations Pattern**, and the **Polymorphism Pattern**. (Hint: Recall that we discussed a similar design in class for switching between tax calculators.)

Draw a class diagram for your design on the next page.


RoboCheF
...
...
1
-Controls
<<interface>>
OvenAdapter
+ myCook(...)
ElectroluxAdapter
...
+ myCook(...)
Maytag Adapter
...
+ myCook(...)
WhirlpoolAdapter
...
+ myCook(...)
1
-adaptee
ElectroluxGasOven
...
...
MaytagMicrowave
...
...
WhirlpoolElectricRange
...
...