

ICS 413/613 Midterm 1
February 20, 2002

NAME: _____

Instructions:

- ?? Please answer all questions on a separate sheet of paper.
- ?? Please answer all questions using a PENCIL, not a pen.
- ?? Neatness and legibility counts. If I cannot read/understand your answer, it's wrong.
- ?? Please write your name on all pages (including this one.)
- ?? Please staple your sheets to this paper when turning it in.

1. Please analyze the accompanying Java source code (Averager) for violations of the java style guidelines used in this class. (This includes both the Elements of Java Style Guidelines and the JOSSE additions published on the website). List the rule number violated next to the violation on the Java source code. List only one violation per rule number. For full credit, find at least 5 unique violations.

Some of the many possible violations include:

- ?? *Line 3, Rule 15 (no package)*
- ?? *Line 9, Rule 39 (No javadoc for private member)*
- ?? *Line 16, Rule 29 (use this.n to refer to instance variable)*
- ?? *Line 22, Rule 5 (need newlines)*
- ?? *Line 48, Rule 7 (need white space)*

2. Describe a top-level design for one or more abstract data types that could be used to implement a program to manage the grades for a set of students in a class. Your "top-level design" should describe the set of packages, classes, and methods in the ADTs. For each package, list the set of classes in it. For each class, indicate its visibility and list the set of methods that comprises it. For each method, indicate its visibility, its return type, and the types of its parameters. Do not include JavaDocs. Do not include method bodies. Do not describe the user interface. Your design can and should include a description of the JUnit test classes and methods.

One of the many possible reasonable designs is:

```
package edu.hawaii.grader;  
public class Grade  
    public static Grade A;  
    public static Grade B;  
    public static Grade C;  
    public static Grade D;  
  
public class Student  
    public Student(String)
```

```
public class Assignment
    public Assignment(String)
```

```
public class StudentAssignmentGrade
    public StudentAssignmentGrade(Student, Assignment, Grade)
    public Grade getGrade()
    public Assignment getAssignment()
    public Student getStudent()
```

```
public class GradeList
    public GradeList()
    public void add(StudentAssignmentGrade)
    public void delete(StudentAssignmentGrade)
    public StudentAssignmentGrade find(Student)
    public StudentAssignmentGrade[] find(Assignment)
    public StudentAssignmentGrade[] find(Grade)
```

```
public class TestGradeList
    public void testEmptyGradeList()
    public void testAddOneGrade()
    public void testAddManyGrades
    public void testAddDuplicateGrade
    public void testFindGrade()
    public void testDeleteGrade()
```

3. The Stack system is constructed so that when the user executes 'java -jar stack.jar', the system prints out a line indicating the day that the jar file was constructed. Please describe how the Stack source code and Ant build system interact to accomplish that goal. You can refer to the line numbers in the printed versions of build.xml and Stack.java in your answer.

Build.xml:18: The "release" property is defined.

Build.xml:19: The file containing the token to be filtered (Stack.java) is defined.

Build.xml:43: The token @release@ is defined.

Build.xml:55: Stack.java is touched, guaranteeing that its tokens will be filtered during copy.

Build.xml:56: A copy of Stack.java is made, during which its @release@ token is replaced by the value of the release property in line 15.

Build.xml:76: The manifest is set for the jar file, setting main-class to edu.hawaii.stack.Stack.

Stack.java:53: The main() method prints the value of release, which was compiled with the time that the prepare target was last executed.

4. Closed-source software can violate one or more of the JOSSE prime directives and still be commercially successful. Pick one of the prime directives and describe why it might not be important in a particular closed-source development setting.

In a closed source setting, the set of developers working on the system might be small and fixed and working together. This has the potential to make the documentation requirements for Prime Directive number 3 unnecessary for success.

5. Describe a design scenario in which you would choose to use a Java interface rather than an abstract class, and defend your decision.

Any design scenario in which multiple inheritance is required leads to the need to choose Java interfaces, since abstract classes support only single inheritance.

6. (613 only) Pfleeger identifies the following types of faults (p. 333): algorithmic, syntax, precision, documentation, overload, boundary, coordination, performance, recovery, hardware, and procedure. Briefly discuss the extent to which the current JOSSE software development process and tools support discovery of each of these fault types.

- ?? Algorithmic faults are faults in the processing steps, which can be uncovered by JUnit tests.*
- ?? Syntax faults are generally caught by the Java compiler during the build.*
- ?? Precision faults occur when the result is not computed to a sufficient degree of accuracy. The current process does not directly support discovery of these fault types.*
- ?? Documentation faults are partially discovered through the JRefactory and Checkstyle tools.*
- ?? Overload (data structure overflow) and boundary (load limits exceeded) faults are not directly supported by JOSSE at this time. (Wait until next semester).*
- ?? Discovery of coordination faults due to improperly synchronized threads are not currently supported by JOSSE.*
- ?? Discovery of performance faults (inadequate speed) is not currently directly supported by JOSSE.*
- ?? Recovery faults are partially supported by JUnit.*
- ?? Hardware faults are not supported.*
- ?? Procedure faults are partially supported through the Ant build process.*