# COMP 401 Fall 2013 Midterm

## Tuesday, Oct 15, 2013, 11:00-12:15

## Instructions

1. Please spread out and try and sit in alternate seats.
2. This is a closed book exam. You will not be penalized for errors in Java syntax.
3. Write on the exam itself.
4. Write on a blank page if there is not enough space to solve a problem.
5. There are:

* 10 numbered pages including this one, and any marked blank pages.
* 3 questions.
* 75 possible points. Point values appear in brackets next to each question.

1. You have 75 minutes.
2. You are not required to comment any code you write, but may get partial credit if you write appropriate comments but incorrect code.
3. If you need to make any assumptions to clarify a problem, *write your assumptions down.* Only reasonable assumptions get full credit.
4. Please inform the proctor of anything in the exam that you think is a mistake.
5. Your code will be evaluated not only for correctness, but also for time and space efficiency and *style*.
6. You cannot use any Java capabilities not covered in class.
7. To answer questions about some piece of code given here, you can mark the code directly.
8. If you do not understand some English word, do not hesitate to ask the proctor. Naturally, you are expected to know the computer science terms defined in class.

Name (in Capitals)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pledge: I have neither given nor received unauthorized aid on this exam.

(signed)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For survey purposes, please indicate the time at which you turned in the exam.

\_\_\_\_\_\_\_\_\_\_\_

Please do not write below

1. \_\_\_\_\_/22 2. \_\_\_\_\_/37 3. \_\_\_\_\_/16

Total: \_\_\_\_\_/ 75

1. [22 pts.] Understanding Programs

Consider the following classes for describing and using regular and extra credit features in an assignment. All of the code related to this problem is located in the same package.

**public** **class** AGradingFeature {

**public** String name;

**int** score;

**public** AGradingFeature() {

score = 100; // give max points

}

**public** AGradingFeature(String aName, **int** aScore) {

name = aName;

score = aScore;

}

**public** String getName() {**return** name;}

**public** **int** getScore() {**return** score;}

}

**public** **class** AnExtraCreditFeature {

String name;

**int** score, extraCreditScore;

**public** AnExtraCreditFeature (String aName, **int** aScore, **int** anExtraCreditScore) {

name = aName;

score = aScore;

extraCreditScore = anExtraCreditScore;

}

**public** AnExtraCreditFeature () { score = 100; }

**public** String getName() {**return** name;}

**public** **int** getScore() {**return** score;}

**public** **int** getExtraCreditScore() {**return** extraCreditScore; }

**public** **boolean** getExtraCredit() {**return** **true**; }

}

**public** **class** TotalScoreComputer {

**public** **static** **int** totalScore(Object anObject) {

**if** (anObject **instanceof** AGradingFeature) {

AGradingFeature gradingFeature = (AGradingFeature) anObject;

**return** gradingFeature.getScore();

} **else** **if** (anObject **instanceof** AnExtraCreditFeature) {

AnExtraCreditFeature extraCreditFeature = (AnExtraCreditFeature) anObject;

**return** extraCreditFeature.getScore() +

extraCreditFeature.getExtraCreditScore();

} **else** **return** -1; // error condition

}

}

**public** **class** GradingFeatureUser {

**static** Object[] *gradingFeatures* = {

**new** AGradingFeature(),

**new** AGradingFeature("No Magic Numbers", 92),

**new** AnExtraCreditFeature(),

**new** AnExtraCreditFeature("No Manual Dispatch", 70, 5)

};

**public** **static** **void** main (String[] args) {

System.*out*.println(((AGradingFeature) *gradingFeatures*[0]).getName());

System.*out*.println(((AGradingFeature) *gradingFeatures*[0]).getScore());

System.*out*.println(((AGradingFeature) *gradingFeatures*[1]).getName());

System.*out*.println(((AnExtraCreditFeature) *gradingFeatures*[3]).getExtraCredit());

System.*out*.println(TotalScoreComputer.*totalScore*(*gradingFeatures*[1]));

System.*out*.println(TotalScoreComputer.*totalScore*(*gradingFeatures*[3]));

}

}

1. [2pts] Identify an instance variable in the code above.
2. [2pts] Identify an instance method in the code.
3. [2 pts] Identify a constructor in the code.
4. [2 pts] Identify a computed property in the code.
5. [2 pts] Identify a collection in the code.
6. [12 pts] Give the value output by each println() of the main method above. You can list each value below or next to the associated println(). We are asking for the value rather than the actual output because you may not know how some value is printed.
7. [37 pts.] Improving Program Style

Your task in this question is to rewrite the code from question 1 (partly reproduced below), improving its style. Your rewrite must follow all rules you have learned in this class. In addition, it must meet the two constraints given below.

**public** **class** AGradingFeature {

**public** String name;

**int** score;

**public** AGradingFeature() {

score = 100; // give max points

}

**public** AGradingFeature(String aName, **int** aScore) {

name = aName;

score = aScore;

}

**public** String getName() {**return** name;}

**public** **int** getScore() {**return** score;}

}

**public** **class** AnExtraCreditFeature {

String name;

**int** score, extraCreditScore;

**public** AnExtraCreditFeature (String aName, **int** aScore, **int** anExtraCreditScore) {

name = aName;

score = aScore;

extraCreditScore = anExtraCreditScore;

}

**public** AnExtraCreditFeature () { score = 100; }

**public** String getName() {**return** name;}

**public** **int** getScore() {**return** score;}

**public** **int** getExtraCreditScore() {**return** extraCreditScore; }

**public** **boolean** getExtraCredit() {**return** **true**; }

}

**public** **class** TotalScoreComputer {

**public** **static** **int** totalScore(Object anObject) {

**if** (anObject **instanceof** AGradingFeature) {

AGradingFeature gradingFeature = (AGradingFeature) anObject;

**return** gradingFeature.getScore();

} **else** **if** (anObject **instanceof** AnExtraCreditFeature) {

AnExtraCreditFeature extraCreditFeature = (AnExtraCreditFeature) anObject;

**return** extraCreditFeature.getScore() +

extraCreditFeature.getExtraCreditScore();

} **else** **return** -1; // error condition

}

}

(1) You should still have regular and extra credit functionality separated into two different classes. All of the constructors and public methods already in AGradingFeature and AnExtraCreditFeature should also be included in the improved versions of these classes. You may add additional methods as needed.

(2) You should still have a main class that performs the same tasks: stores the same information in four elements of an array and prints the same information in the main method. You must decide if you should distribute the functionality of TotalScoreComputer in the other three classes (by adding new methods) or keep it in a separate class

All code except the main class is reproduced above. All code except the two grading feature classes is reproduced on the next filled page. Use the next two blank pages and, if necessary, the last page of the exam, to do the rewrite. You are free to mark the given code to indicate how it should be changed. If there is more than one way to improve the code, use the one you think is preferable. You are welcome but not required to explicitly identify each problem you fix and the nature of your solution to it. You may get partial credit if you point out a problem but do not fix it properly.

Code reproduction.More space for rewrite.

**public** **class** TotalScoreComputer {

**public** **static** **int** totalScore(Object anObject) {

**if** (anObject **instanceof** AGradingFeature) {

AGradingFeature gradingFeature = (AGradingFeature) anObject;

**return** gradingFeature.getScore();

} **else** **if** (anObject **instanceof** AnExtraCreditFeature) {

AnExtraCreditFeature extraCreditFeature = (AnExtraCreditFeature) anObject;

**return** extraCreditFeature.getScore() +

extraCreditFeature.getExtraCreditScore();

} **else** **return** -1; // error condition

}

}

**public** **class** GradingFeatureUser {

**static** Object[] *gradingFeatures* = {

**new** AGradingFeature(),

**new** AGradingFeature("No Magic Numbers", 92),

**new** AnExtraCreditFeature(),

**new** AnExtraCreditFeature("No Manual Dispatch", 70, 5)

};

**public** **static** **void** main (String[] args) {

System.*out*.println(((AGradingFeature) *gradingFeatures*[0]).getName());

System.*out*.println(((AGradingFeature) *gradingFeatures*[0]).getScore());

System.*out*.println(((AGradingFeature) *gradingFeatures*[1]).getName());

System.*out*.println(((AnExtraCreditFeature) *gradingFeatures*[3]).getExtraCredit());

System.*out*.println(TotalScoreComputer.*totalScore*(*gradingFeatures*[1]));

System.*out*.println(TotalScoreComputer.*totalScore*(*gradingFeatures*[3]));

}

}

1. [16 pts] WRITING CODE

Write a function that trims a string, that is, takes a string as an argument and returns a copy of this string with all leading and trailing space characters removed. The following table shows some example arguments and return values.

|  |  |
| --- | --- |
| **Argument** | **Return Value** |
| " Hello world" | "Hello world" |
| " Hello world " | "Hello world" |
| "Hello world " | "Hello world" |
| "Hello world" | "Hello world" |
| “ 12#$, \*hy “ | “12#$, \*hy“ |
| " " | "" |
| "" | "" |

As in your assignments, you should implement this function using only Java features you have learned in class. The relevant String methods are described below.

public char **charAt**(int index)

Returns the char value at the specified index.

public int **length**()

Returns the length of this string.

public [String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) **substring**(int beginIndex,

int endIndex)

Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex.

Examples:

"hamburger".substring(4, 8) returns "urge"

"smiles".substring(1, 5) returns "mile"

"smiles".substring(1, 1) returns ""

The header of the method is given below. Fill the body of the method.

**public** **static** String trim(String aString) {

}