# Computing and Algorithms I

# Project 3

## CS–101

## Summer, 2013

This is a 25 point assignment. It must be turned in by the beginning of class on Thursday of week 6 (August 22) for credit.

This assignment is an exercise in converting a UML diagram of a class, with algorithms for each method of the class, into code for a class. Note that you are not writing an application program. I will be testing your code with my class which contains a main method. We will start with two examples of UML diagrams and algorithms which have been coded, then there will be a section giving a UML diagram of a class which you will convert to correct Java code.

## Example 1 Counter with Reset and Unreset

### UML class design

Legend

| | |
|---|---|
| + | public |
| − | private |
| | package |
| # | protected |

| | |
|---|---|
| + | Counter |
| − | count:int |
| − | lastCount:int |
| + | Counter() |
| + | increment() |
| + | reset() |
| + | unreset() |
| + | getCount():int |
| + | toString():String |

### Algorithms

Counter()
    count ← 0
    lastCount ← 0

increment()
    count ← count + 1

reset()
    lastCount ← count
    count ← 0

unreset()
    count ← lastCount
    lastCount ← 0

getCount()
    return count

toString()
    return "Counter state: count = " + count + " lastCount = " + lastCount

**Java Code**

```
/**
 *  Counter class for creating Counter objects.
 *  A counter starts at 0 and counts by increments of 1 until reset
 *  File: Counter.java
 *  Designer: Dr. Vineyard
 *  Implementor: Dr. Vineyard
 *  Organization: CS101
 *
 * Data Table
 *
 *  Variable             Usage
 *  --------             -----
 *  count                integer counted to
 *  lastCount            set to value int count on reset()
 *                         used to restore count on unreset()
 */

public class Counter
{
    private int count;
    private int lastCount;

    /**
     * Counter is the constructor.  It takes no arguments
     *
     * algorithm:
     * count <-- 0
     * lastCount <-- 0
     */
    public Counter()
    {
        count = 0;
        lastCount = 0;
    }

    /**
     * increment adds 1 to the count of the counter
     *
     * algorithm:
     * count <-- count + 1
     */
    public void increment()
    {
        count = count + 1;
```

```
}

/**
 * reset puts count back to 0
 *
 * algorithm:
 * lastCount <-- count
 * count <-- 0
 */
public void reset()
{
    lastCount = count;
    count = 0;
}

/**
 * unreset undoes the effect of reset
 *
 * algorithm:
 * count <-- lastCount
 *lastCount <-- 0
 */
public void unreset()
{
    count = lastCount;
    lastCount = 0;
}

/**
 * getCount returns the value of the counter
 *
 * algorithm:
 * return count
 */
public int getCount()
{
    return count;
}

/**
 * toString() returns a String representation of the state of the object
 *
 * algorithm:
 * return count and lastcount in a String
 */
```

```
    public String toString()
    {
        return "Counter state: count = " + count + " lastCount = " + lastCount;
    }
}
```

## Example 2 Student

### UML class design

| Legend | |
|---|---|
| + | public |
| − | private |
| | package |
| # | protected |

| + | Student |
|---|---|
| − | firstName:String |
| − | lastName:String |
| + | MAX_QUIZ_SCORE=20:int |
| − | totalQuizScore:int |
| − | numberOfQuizes:int |
| + | Student(firstName:String, lastName:String) |
| + | addQuiz(score:int) |
| + | getScore():int |
| + | getAverage():double |
| + | getName():String |
| + | toString():String |

### Algorithms

```
Student(firstName, lastName)
    this.firstName ← firstName
    this.lastName ← lastName
    totalQuizScore ← 0
    numberOfQuizes ← 0
addQuiz(score)
    totalQuizScore ← totalQuizScore + score
    numberOfQuizes ← numberOfQuizes + 1
getScore()
    return totalQuizScore
getAverage()
    return 1.0∗ totalQuizScore/numberOfQuizes
getName()
    return firstName + " " + lastName
toString()
    return firstName + " " + lastName + " has total quiz score of " + totalQuizScore +
            " in " + numberOfQuizes + " quizes."
```

**Java Code**

```
/**
 *  Student class for creating Student objects.
 *  A Student consists of a name and some quiz score information
 *  File: Student.java
 *  Designer: Dr. Vineyard
 *  Implementor: Dr. Vineyard
 *  Organization: CS101
 */

/**
 * Data Table
 *
 *  Variable            Usage
 *  --------            -----
 *  name                String data for student's name
 *  MAX_QUIZ_SCORE      constant value, highest grade on a quiz
 *  totalQuizScore      sum of all the quiz scores taken by student
 *  numberOfQuizes      count of quizes taken by student
 */

public class Student
{

    private String firstName;
    private String lastName;
    public final static int MAX_QUIZ_SCORE = 20;
    private int totalQuizScore;
    private int numberOfQuizes;

    /**
     * Student constructor initializes name and quiz values
     *
     * algorithm:
     * this.firstName <-- firstName
     * this.lastName <-- lastName
     * totalQuizScore <-- 0
     * numberOfQuizes <-- 0
     */
    public Student(String firstName, String lastName)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        totalQuizScore = 0;
        numberOfQuizes = 0;
```

```
}

/**
 * addQuiz adds a quiz score to total and increments number of quizes
 *
 * algorithm
 * totalQuizScore <-- totalQuizScore + score
 * numberOfQuizes <-- numberOfQuizes + 1
 *
 */
public void addQuiz(int score)
{
    totalQuizScore = totalQuizScore + score;
    numberOfQuizes = numberOfQuizes + 1;
}

/**
 * getScore returns the accumulated quiz score
 *
 * algorithm
 * return totalQuizScore
 *
 */
public int getScore()
{
    return totalQuizScore;
}

/**
 * getAverage computes and returns the average quiz score
 *
 * algorithm
 * return 1.0 * totalQuizScore / numberOfQuizes
 *
 */
public double getAverage()
{
    return 1.0 * totalQuizScore / numberOfQuizes;
}

/**
 * getName returns the name of the student
 *
 * algorithm
 * return firstName, lastName
```

```
     *
     */
    public String getName()
    {
        return firstName + " " + lastName;
    }

    /**
     * toString returns current state of the student object
     *
     * algorithm
     * return firstName, lastName, totalQuizScore, and numberOfQuizes as String
     *
     */
    public String toString()
    {
        return firstName +  "  " + lastName + " has total quiz score of " +
               totalQuizScore + " in " + numberOfQuizes + " quizes.";
    }
}
```

# UML Design for Student Implementation

## UML class design

Legend

| | |
|---|---|
| + | public |
| − | private |
| | package |
| # | protected |

| | |
|---|---|
| + | Course |
| − | name:String |
| − | number:int |
| + | MAX_STUDENTS=40:int |
| − | numberOfStudents:int |
| − | building:String |
| − | roomNumber:int |
| + | Course(name:String, number:int, building:String, room:int) |
| + | addStudents(students:int) |
| + | dropStudents(students:int) |
| + | getCourseNumber():int |
| + | getNumberOfStudents():int |
| + | getCourseName():String |
| + | getBuilding():String |
| + | getRoomNumber():int |
| + | toString():String |

## Algorithms

Course(name, number, building, room)
    this.name ← name
    this.number ← number
    numberOfStudents ← 0
    this.building ← building
    roomNumber ← room

addStudents(students)
    numberOfStudents ← numberOfStudents + students

dropStudents(students)
    numberOfStudents ← numberOfStudents − students

getCourseNumber()
    return number

getNumberOfStudents()
    return numberOfStudents

getCourseName()
    return name

getBuilding()
    return building

getRoomNumber()
    return roomNumber

toString()
    return name + " Course Number " + number + " Number of Students " + numberOfStudents + " Building " + building + " Room Number " + roomNumber

## Deliverables

Convert the Course design into Java code using comments similar to the comments in the examples. Note that you, and not the instructor, are the implementor for this code. When your code is complete and compiles correctly, print the code, and submit the Java program (Course.java after zipping) using blackboard. Turn in the printout at the beginning of class on August 22, submit the code as a zip file on blackboard before the beginning of class on August 22.