# CSCI 141 Test 2 – Spring 2015

Create one Eclipse project and answer the following questions. Make as many classes as you need. Export your project and submit to Blackboard before class ends. Note, none of your code needs Scanner or any user input. Use comments for answers that are not valid code. Points total 50pts.

(1pt.) For the following array, what value is `vals[2]` ?

```
boolean[] vals = {false, true, false, true, false};
```

(6pts.) Finish this chunk of code so that the array `intseq` is filled with values 1 to 1000 (in increasing order). Do not convert the code into a "`for`" loop.

```
int[] intseq = ...
int i = ...
while(...) {
     intseq[...] = ...
     ...
}
```

(4pts.) Given this function in the `Arrays` class (`import java.util.Arrays`):

```
// Copies the specified range of the specified array
// into a new array.
// "from" is start (inclusive), "to" is 1+end (exclusive)
char[] copyOfRange(char[] original, int from, int to)
```

finish the code below so that it works as described. Feel free to test it yourself.

```
char[] symbols = {'q', 'w', 'e', 'r', 't', 'y'};
char[] middleSyms = Arrays.copyOfRange(...); // finish this
// middleSyms now equals {'e', 'r', 't'}
```

(15pts.) Write a function called `sumArrays` that adds two double arrays. It returns the resulting array (and does not modify the originals). Check if the two input arrays have the same length; if they do not, print an error message and return either of the input arrays. Examples:

```
double[] xs = {5.5, 1.0, 2.3, 0.4};
double[] ys = {3.2, 9.1, 2.2, 0.0};
double[] sum = sumArrays(xs, ys);
// sum now equals {8.7, 10.1, 4.5, 0.4}

double[] xs = {5.5, 1.0, 2.3, 0.4};
double[] ys = {3.2, 9.1};
double[] sum = sumArrays(xs, ys);
// prints an error message, arrays are not the same length
// sum now equals xs or ys (doesn't matter which)
```

(2pts.) Assuming a class has this constructor:

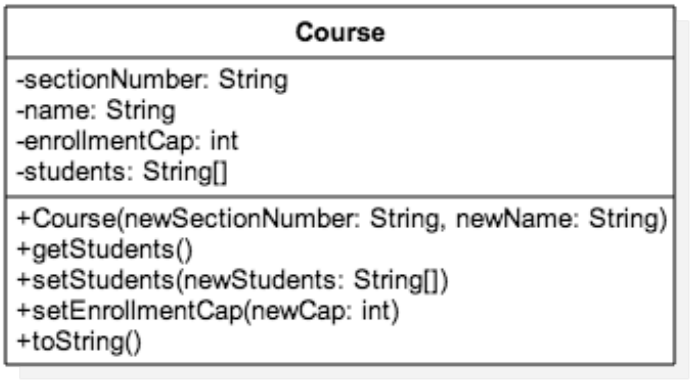
```
Foo(int bar, char baz)
```

write two lines of code that create *two* different objects of type `Foo`.

(1pt.) Assuming a class `Quux` has a method `void doThatThing(double withThisThing)`, and you have an object of type `Quux` with the name `myQuux`, write one line of code that executes `doThatThing` to the `myQuux` object.

(1pt.) If a class called `Configuration` had a public field `int screenWidth` (among others), and you had a `Configuration` object called `config`, write one line of code, without using a class method, that would set the screen width to 2048.

(20pts.) Given this UML diagram representing the `Course` class,




write all the code for the class, for only those methods shown in the UML diagram (so, e.g., there is no `setSectionNumber` method). The return types of the methods are not shown, but you should be able to figure them out. Notice that the *constructor* does not allow you to set the students or enrollment cap; this is by design. Write the code so that the code below (in `main`) works:

```
Course csci141 = new Course("CSCI141", "Intro to CS I");
csci141.setEnrollmentCap(20);
Students[] studs = {"Lisa", "Jake", "Ananda", "Yves"};
csci141.setStudents(studs);
System.out.println("Here is the course info: " + csci141);
```

That last line should cause the following text to print (notice it includes the # of students and shows the enrollment cap: "4/20"). Obviously, it's the `toString` method that creates the text about the course:

```
Here is the course info: CSCI141 – Intro to CS I – 4/20
students are enrolled
```