7-5

CS 1401, Exam #4, MW 9-10:20 version

57/60 = 95/100

Date: Wednesday, November 27, 2013
Name (please type legibly, ideally in block letters): [redacted]

On November 27, 1852, Ada Lovelace, the daughter of Lord Byron, died.

10/10

1. Describe, in detail, how Ada Lovelace contributed to computing.

Ada Lovelace contributed to computing by becoming the first computer programmer. Lovelace was an assistant to Charles Babbage who came up with the idea for a computational device known as the Analytical Engine. Although the device was not a success, Lovelace was the one who designed the programs that would run it.

10/10

2a. The programmer's productivity can be measured by the average number of lines of code that the programmer produces per day. Write a method named *productivity* that, given the number of lines of code produced by the programmer during a certain period of time and the duration *d* of this period, returns the programmer's productivity. In your method, throw an exception to take into account situations when the duration is zero.

2b. Throw an additional exception when the number of lines is negative.

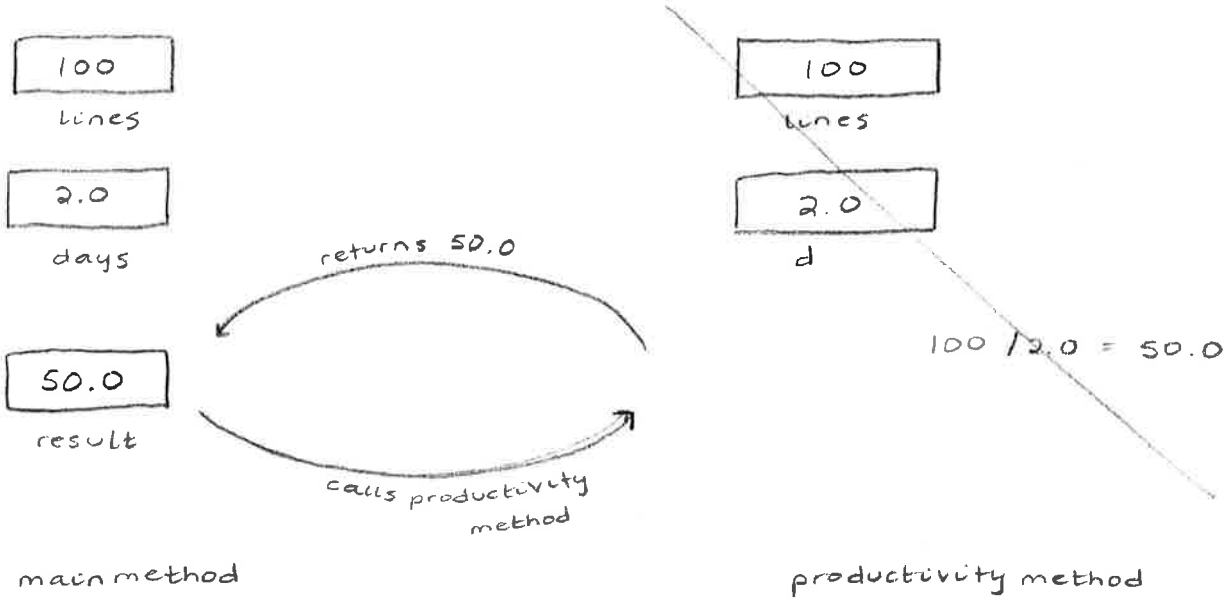
2c. Call (invoke) your method *productivity* in the *main* method to compute the productivity of a programmer. Use try-catch in your main method to catch the two exceptions and print the corresponding error messages. On the example when a programmer coded 100 lines in 2 days, trace, step by step, how the computer will perform the needed computations, and check that the result is indeed correct.

a.)
```
public double productivity(int lines, double d){
    if(d==0){
      throw new ArithmeticException("Cannot divide by zero.");}
    return (lines/d);}
```

b.)
```
public double productivity(int lines, double d){
    if (d==0){
     throw new ArithmeticException("Cannot divide by zero.");}
    if (lines<0){
     throw new IllegalArgumentException("Lines must be positive.");}
    return (lines/d);}
```

c.)
```
public static void main(String[] args){
    int lines=100; double days=2.0;
    try{ double result = productivity(lines, days);}
    catch(ArithmeticException ex1){
    System.out.println("Error: Cannot divide by a negative number.");}
    catch(IllegalArgumentException ex2){
    System.out.println("Error: number of lines must be positive.");}
```

Tracing on back →


100
lines
2.0
days
50.0
result
returns 50.0
calls productivity method
100
lines
2.0
d
100 / 2.0 = 50.0
main method
productivity method

10/10

3a. Define a class *Programmer* whose objects are different programmers. The description of each programmer should contain his/her name, the number of lines of code this programmer wrote, and the amount of time (in days) that the programmer works for the company. Your class should contain a constructor method, get- and set-methods, and a method for computing the programmer's productivity.

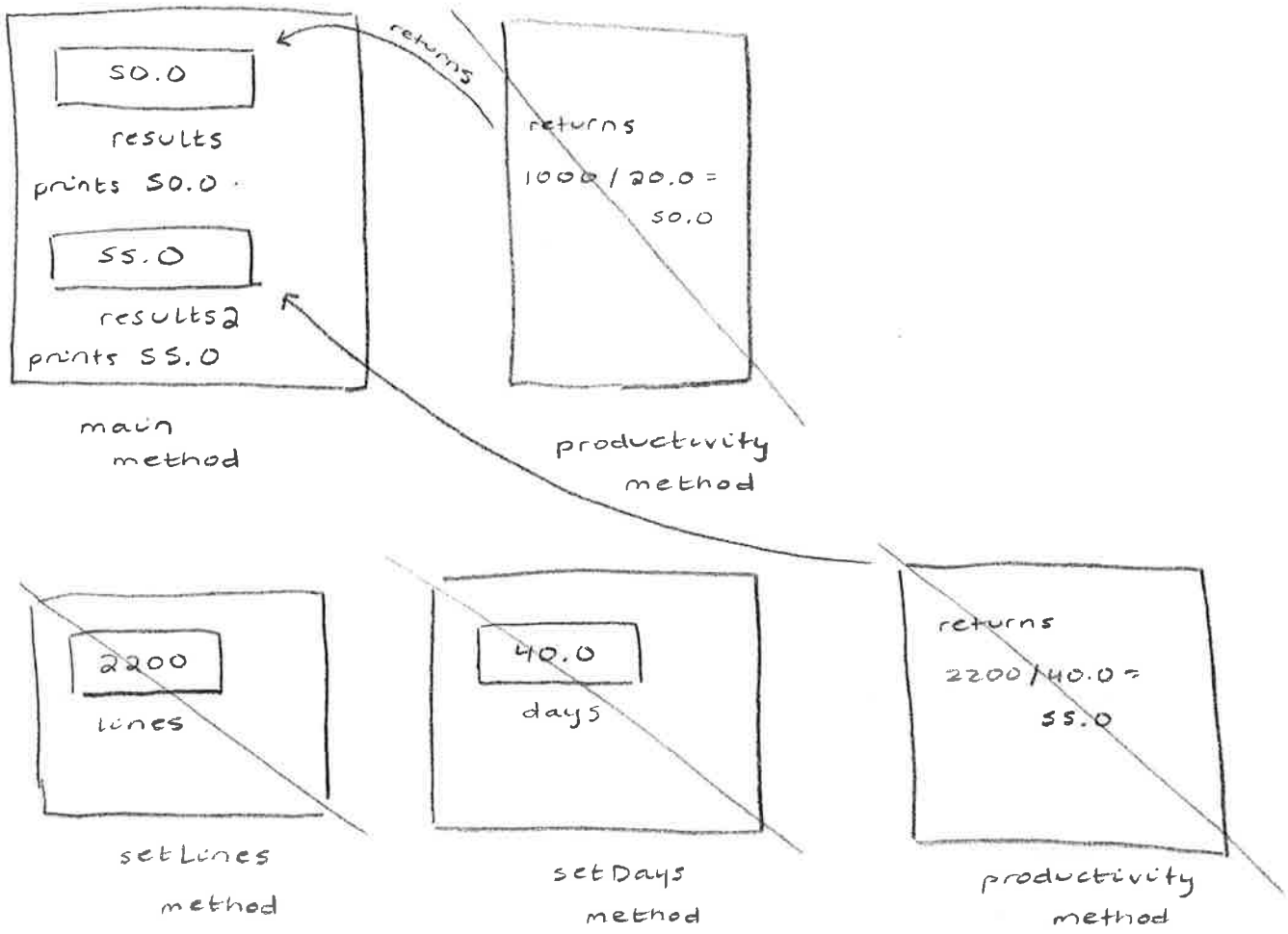
3b. Use your class in the *main* method to define a new object *mario* of type *Programmer*; assume that Mario programmed 1,000 lines of code in 20 days. Compute and print Mario's productivity. While you were computing, he has programmed 1,200 more lines of code in another 20 days. Use the set-methods to update the values in the corresponding object, and compute and print his updated productivity.

3c. Trace your program step-by-step.

a)
```
public class Programmer {
    private String name; private int lines; private double days;
    public Programmer (String n, int li, double d){
    name = n; lines = li; days = d; }
    public String getName() { return name; }
    public int getLines() { return lines; }
    public double getDays() { return days; }
    public void setName (String n) { name = n; }
    public void setLines (int li) { lines = li; }
    public void setDays (double d) { days = d; }
    public double productivity() { return lines / days; } }
```

b.)
```
public static void main(String [] args) {
    Programmer mario = new Programmer ("Mario", 1000, 20.0);
    double results = mario.productivity();
    System.out.println (results);
    mario.setLines (2200);
    mario.setDays (40.0);
    double results2 = mario.productivity();
    System.out.println (results2);
```

c.) Tracing on back →


50.0
results
prints 50.0
55.0
results2
prints 55.0
returns
returns
1000 / 20.0 =
50.0
main
method
productivity
method

2200
lines
40.0
days
returns
2200 / 40.0 =
55.0
setLines
method
setDays
method
productivity
method

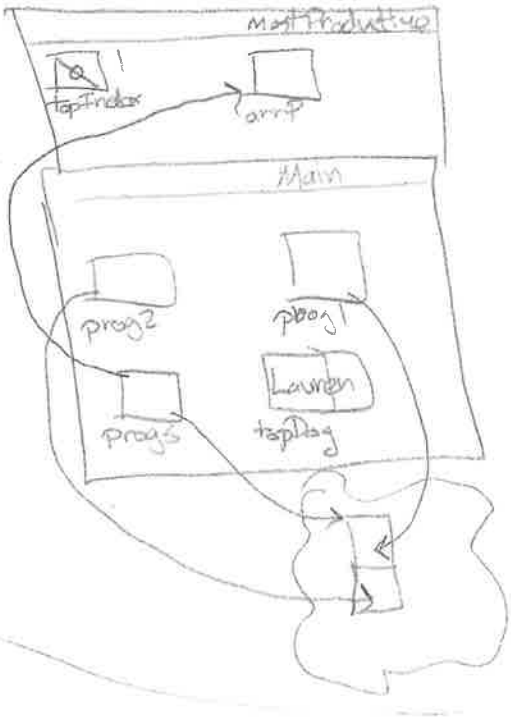
4a. Write a method that, given an array of programmers, returns the name of the most productive programmer.

4b. Test your method in the main program, by applying it to two programmers: Mario and Lauren; assume that Lauren programmed 3,000 lines of code in 15 days. Trace the resulting code.

4c. Describe general rules for black-box and white-box program testing.

```
public String mostProductive (Programmer [] arrP)
{int topIndex = 0;
  for (int i = 1 ; i < arrP.length , i++ )
    { if (arrP[topIndex].getProductivity() > arrP[i].getProductivity)
       { topIndex = i ;}
    } // end for
  return arrP[topIndex].getName ;}
```

```
public static void main(String[] args)
{ Programmer prog1 = new Programmer ("Mario", 1000, 20 );
  Programmer prog2 = new Programmer ("Lauren", 3000, 15);
  Programmer [] progs = {prog1, prog2};
  String topDog = mostProductive (progs);}
```



Testing Rules

- Boundary values
- Random values
- known problem values (like 0)
- Test the extremes of the possible input ranges

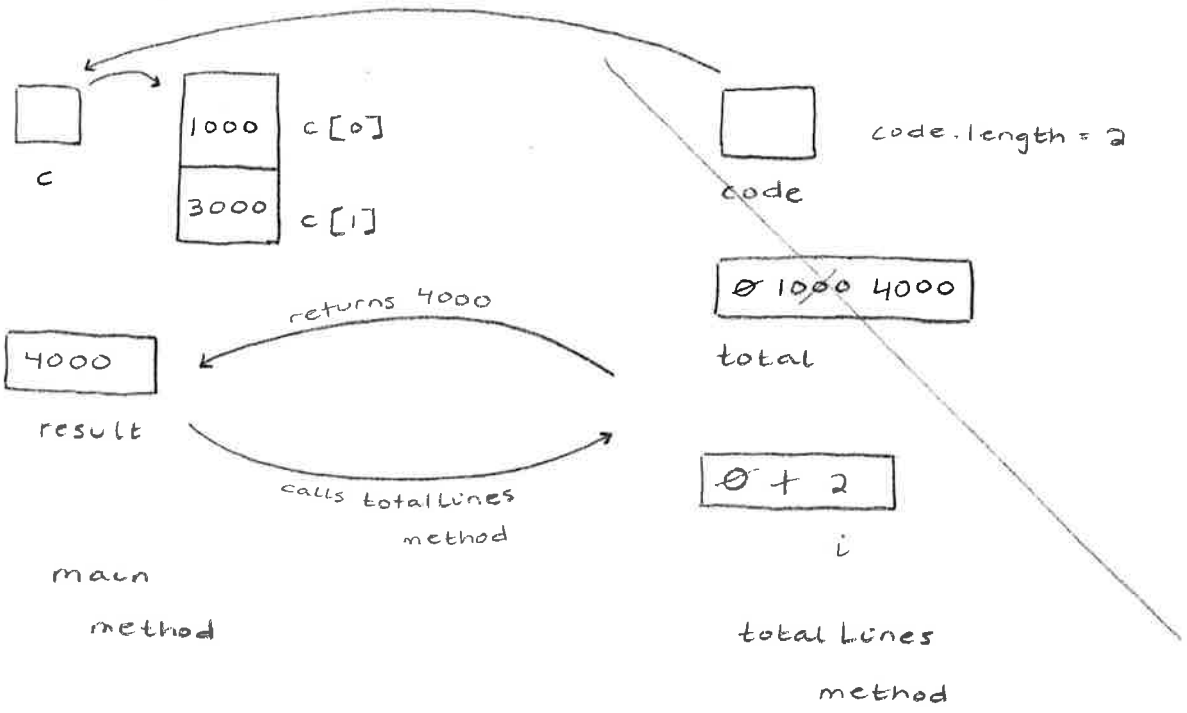
10/10

5a. Write a method that, given the array *code* of lines of code written by different programmers, returns the total number of lines of code. 5b. In the main method, apply your code to the array consisting of two values 1,000 and 3,000; trace your code, step-by-step, and check that your method returns the correct number 1,000 + 3,000 = 4,000.

a.)
```
public int totalLines(int[] code){
  int total = 0;
  for(int i = 0; i < code.length; i++){
   total += code[i];}
  return total;}
```

b.)
```
public static void main(String[] args){
  int[] c = {1000, 3000};
  int result = totalLines(c);}
```

Trace:



10/10

6. Give at least two reasons why cheating on a test is not ethical. For example, is it fair to others who do not cheat? Is it fair to those who hire a former student assuming, from his/her good grades, that this student knows how to program? Is it fair to the customers who will rely on the code written by this student? Give detailed explanations.

1.) Cheating on a test is not ethical because the student who cheats does not actually learn the necessary material. This not only affects the student, who is depriving himself/herself of essential knowledge, but it also affects customers who may rely on code written by this student. It is not fair for a customer to be dependent on a program that may contain damaging errors due to a student's lack of skill.

2.) It is also not ethical to cheat on an exam because it is not fair to those who actually took the time to study in order to get a good grade. It is not right for someone who took the 'easy route' to achieve the same grade as a student who spent hours studying.