49/50 = 98/100

6-8

CS 1401, Exam #3, MW 12-1:20 pm version

Date: Wednesday, November 6, 2013
Name (please type legibly, ideally in block letters):
Jose Rojas

On November 6, 1528, conquistador Alvar Nunez Cabeza de Vaca became the first European to set foot in what is now Texas.

1a. Many conquistadors were paid a salary by Spain. Write a method named *earning* that, given the annual salary *s* in doubloons and the number of years *y* the conquistador served, returns the amount of doubloons that this person received during his lifetime.

10/10

1b. Call (invoke) your method *earning* in the *main* method to compute the amount of money received by Alvar Nunez Cabeza de Vaca who received 50 doubloons per year for 15 years. You do not need to write the entire *main* method, just the part that assigns values to the corresponding variables *salary* and *years* and calls your method.

1c. Trace, step by step, how the computer will perform the needed computations, and check that the result is indeed correct.

1a.)
```
public double earning(double s, int y) {
    return s*y;
}
```

1b.)
```
int years = 15;
double salary = 50.0;
double lifetimeEarnings = earning(salary, years);
```

1c.)

1 [15] years

2 [50.0] salary

3 [7500] lifetimeEarnings → earning(salary, years) [15*50.0] return

2a. Define a class *Conquistador* whose objects are different conquistadors. The description of each conquistador should include his name, his birth year, and the age at which he came to the Americas. Your class should contain a constructor method, get- and set-methods, and a method for computing the year when this conquistador arrived in the Americas.

2b. Use your class in the *main* method to define a new object *nunez* of type *Conquistador* describing Alvar Nunez Cabeza de Vaca. He was known to be born in 1490, and he arrived in the New World at the age of 37. Compute and print the year when he arrived in the New World. Then, replace the year 1490 with the corrected year 1488 provided by some historians, and compute and print the corrected arrival year.

10/10

2c. Trace your program step-by-step.

2a)

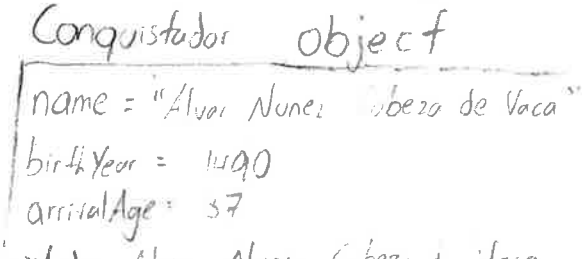
```
public class Conquistador {
    private String name;
    private int birthYear;
    private int arrivalAge;

    public Conquistador(String name, int birthYear, int arrivalAge){
        this.name = name;
        this.birthYear = birthYear;
        this.arrivalAge = arrivalAge;
}
public String getName(){
        return name;
}
public int getBirthYear(){
        return birthYear;
}
public int getArrivalAge(){
        return arrivalAge;
}
```

```
public int getArrivalYear(){
    return birthYear + arrivalAge;
}
public void setName(String name){
    this.name = name;
}
public void setBirthYear(int birthYear){
    this.birthYear = birthYear;
}
public void setArrivalAge(int arrivalAge){
    this.arrivalAge = arrivalAge;
}
}
```

2b)

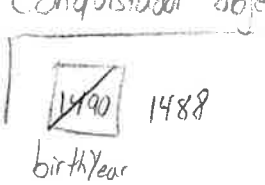
```
Conquistador nunez = new Conquistador("Alvar Nunez Cabeza de Vaca", 1490, 37)
System.out.println("Conquistador " + nunez.getName() + "arrived to the New World"
                   + " on " + nunez.getArrivalYear());
nunez.setBirthYear(1488);
System.out.println("Conquistador " + nunez.getName() + "arrived to the New World" +
                   " on " + nunez.getArrivalYear());
```

2.)



2- Prints: Conquistador Alvar Nunez Cabeza de Vaca arrived to the New World on 1527

3-



4- Prints: Conquistador Alvar Nunez Cabeza de Vaca arrived to the New World on 1525

3a. Write a piece of code that, given an array *birthYear* of birth years of famous people and an array *name* of their names, prints the names of those who are exactly 100, 200, 300, etc. years old in 2013 (i.e., whose age is divisible by 100). Assume that the arrays have been declared, initialized, and that they have the same length.

10/10

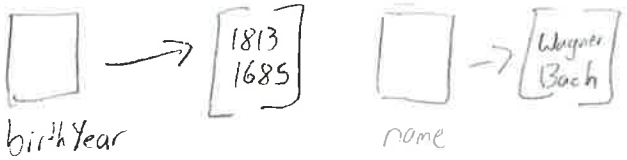
3b. To check the correctness of the code you wrote in Part 3a, write a piece of code that defines a new array *famousBorn* with years 1813 and 1685, and a new array *famousName* with elements Wagner and Bach.

3c. Trace step-by-step how the piece of code you wrote in Part 3a will compute the corresponding ages.

3a)
```
public void OldandFamous(int birthYear[], String name[]) {
    for(int i = 0; i < birthYear.length; i++) {
        int AgeDivisibility = (2013 - birthYear[i]) % 100
        if(AgeDivisibility == 0) {
            System.out.println(name[i]);
        }
    }
}
```

3b)
```
int famousBorn[] = {1813, 1685};
String famousName[] = {"Wagner", "Bach"};
OldandFamous(famousBorn, famousName);
```

3c)



i	AgeDivisibility	birthYear[i]	name[i]
0	0	1813	Wagner
1	28	1685	Bach

Prints: Wagner

4a. Several people claimed to be the first to discover Texas. Write a method that, given an array *t* of years when each of them reaches Texas first and the array *n* of their names, finds the name of the person who arrived in Texas first. Assume the arrays have been declared, initialized, and have the same non-zero length.

4b. To check the correctness of your method, write a piece of code that defines new arrays *inTexas* consisting of 3 elements 1530, 1528, and 1529, and *names* with elements V, N, and C.

4c. Trace step-by-step how the piece of code you wrote in Part 4a will find the name of the first European to arrive in Texas. *Hint:* Nunez (N) is the one.

4a)
```
public String discoverer(int t[], String n[]){
    int chosen = 0;
    for(int i=0; i< n.length; i++){
        if(t[chosen] < t[i]){
            chosen = i;
        }
    }
    return n[chosen];
}
```

4b)
```
int inTexas[] = {1530, 1528, 1529};
String names[] = {"V", "N", "C"};
System.out.println("The first to reach Texas was" + discoverer(inTexas, names));
```

4c)

i	t[chosen]	t[i]	chosen
0	1530	1530	0
1	1530	1528	1
2	1528	1529	1

returns: "N"

5a. Describe what is white-box and black-box testing.

5b. Describe the main rules for testing.

9/10

5a) white-box testing is done when someone who knows the inner details of the code is the one who tests it, and compares them to the expected output

black-box testing is done when the tester is unaware of the inner details of the code and provides input expecting some output.

5b) Main rules are checking the actual output to the expected, and making sure all code paths are tested, ex. if-else possibilities etc.

- random values
- boundary values
- typical values