Computer Science AP
Class Practice 01

Start a new Java project and call is *ClassWork*. Complete the following in that project. By the end of this long worksheet you will have good examples of many object orientated programming concepts – starting off with 'classes'.

Section 01 [Class Creation]

Create a class called *Student*. Students should have
- first name
- student id
- grade

Create a runner class called *StudentRunner*.
In the main method you should
- create a student with name Bob, id set to 12345, grade 10
- create another student with name Jane, id set to 6789, grade 8
- show that you can print out the name of the first student
- show that you can print out the grade of the second student

Create a class called *BankAccount*. BankAccounts should have
- account number
- balance (how much money is in the account)

Create a runner class called *BankAccountRunner*.
In the main method you should
- create a BankAccount and set it's member variables
- change the value of the account's balance
- print out the value of the account's balance

continued...

Section 02 [Adding Constructors]

Add a default constructor (no arguments) to the Student class that
- sets the name to "NA", the student id to -1, and the grade to -1

Test this in your runner by creating a student and then immediately printing out it's three member variables to see if they were set properly.

Add an overloaded constructor to the Student class that
- accepts three arguments
- set the value of name, id, and grade to these arguments

Test this in your runner by using this constructor to create a student. Immediately print out the member variables to make sure it set all of them properly.

Add a default constructor to the BankAccount class that
- sets the account number to -1
- sets the balance to 0

Test this in your runner.

Add a second constructor to the BankAccount class that
- accepts two arguments (first argument account number, second is balance)
- will check the first argument to make sure it is a value between 0 and 1000000. If it is, it will set the bank account number to the value of the argument. If not, it will set the bank account number to -1.
  (this is done to make sure that bank account numbers can only be set to values between 0 and 1000000)
- will check the second argument to make sure it is a value that is zero or larger. If it is, it will set the bank account balance to the value of the argument. If not, it will set the bank account balance to 0.
  (this is done to ensure that bank accounts can only be created with a balance of zero or higher)

Test this in your runner by creating several BankAccounts with the second constructor using a variety of values (some that are good, some that are bad) and then printing out the account number and balance to make sure that your constructor code is validating and setting values.

continued…

Section 03 [Adding Simple Methods]

With the Student class add a method call *showStudentInfo()* that will print out the value of all student variables.

Download and copy the file *Records.java* from the RESOURCES folder. Add this file into your current project. The Records class represents an object that will be used to store 8 grades and provide methods to work with the grades.

Read the Records class. Notice that the class has an array of integers to store 8 student grades. Notice that the constructor is currently set up to randomly fill the array with values to make it easier for us to test out the class.

With the Record class, add the following methods:

showGrades()	prints all eight grades out
showAverage()	calculates and prints out the average of the 8 grades
showBestGrade()	calculates and prints out the best grade from the 8 grades

Section 04 [Methods with Parameters / Return Values]

With the Student class

- create mutator and accessor methods for all private class variables

With the BankAccount class

- create a method called *getBalance* that returns the balance of the bank account
- create a method called *deposit(double amount)* that will add the amount to the balance of the bank account. Amount must be larger than zero or the balance will not be changed.
- create a method called *withdraw(double amount)* that will withdraw the amount from the account. Amount must be a positive value and amount must be less than or equal to the balance of the account (can't withdraw more than you have!).
- create a method called *futureBalance(int years, double rate*) that will calculate how much this bank account balance will be, using simple interest, in the given number of years at the given rate. Example futureBalance(5, 0.07) for 5 years at 7% interest rate.

With the Records Class

- change the three methods showAverage, and showBestGrade so that the methods RETURN the values calculated (instead of printing them out).
- change the name of these two methods to *getAverage()* and *getBestGrade()* so that they describe their purpose more accurately
- write the method *public int getGrade(int index)* that will return the grade at the appropriate index position. If the index position passed in is not a valid value (not in the range 0-7 since there are only 8 grades in array) then return the value -1 .

continued…

Section 05 [toString]

For the BankAccount class, code the *toString* method so that it returns the account number of the account like this: "Bank Account 1234" (1234 should be the actual account number).

For the Student class, code the *toString* method so that it returns the name, id, and grade of the student as one string: "John 12345 9"

Section 06 [Classes in Classes]

Part One
Download the class *Address.java* from the RESOURCES folder. This class represents the address of a person. Add it to your current project.

Go back to your Student class and give each Student an Address called *address*.

In a runner program, create a Student.
Set the address information of the student to 123 Hastings Street, Vancouver, BC, v7h5d8, Canada.

Show you know how to access the address information by printing out the street address and the city of the student.

In your runner, create another Student but don't set any address information. Try to print out the street address of the student. Did it work? If not, why?

Part Two
Give each Student a Record field called *records* – you already have this class in your project from earlier.
In the code of your constructor/s for the Student class, set *records* equal to a new instance of the Record class.
(remember that earlier on you set your Records class to fill its grades array with random numbers for testing purposes so your student will have grades already set!)

In your runner program

- use the *showGrades* method to print out the grades of your test student
- use the *getAverage* method to get and print out the student's average grade
- print out the student's grade for their first course (the grade stored in the first index position of the grades array using two different ways
    a) using the getGrade method you coded in the Record class
    b) grabbing the grade directly from the array itself