From: **Alan Liang** alan.5100@yahoo.com
Subject: Updated assignment06 instructions
Date: October 25, 2016 at 9:32 PM
To: Winnie Liang winnieliang27@yahoo.com

AL

Assignment 6

Due Tuesday 10/25 just before midnight. Expect Assignment 7 that will overlap with this one.

All the classes are in the package `assignment06`.

Generating lots of names for test cases can be hard, so this time we will allow up to 308,915,776 names that are all 6 character with upper case letters "AAAAAA" through "ZZZZZZ"

Consider the class `Question1`:

```
public class Question1 {
   public static String name(int n) {
      if (n >= 26*26*26*26*26*26) return "ZZZZZZ";
      if (n < 0) return "AAAAAA";
      char[] chs = Integer.toString(n, 26).toCharArray();
      char[] chs2 = new char[6];
      System.arraycopy(chs, 0, chs2, 6-chs.length, chs.length);
      for(int i = 0; i < 6; i++) {
         if(chs2[i] == 0) chs2[i] = 'A';
         if(chs2[i] >= '0' && chs2[i] <= '9') chs2[i] = (char)('A' + chs2[i]-'0');
         if(chs2[i] >= 'a' && chs2[i] <= 'p') chs2[i] = (char)('A' + 10 + chs2[i]-'a
');
      }
      return new String(chs2);
   }
   public static void main(String[] args) {
      for(int n = 10000; n < 11000; n++) {
         System.out.println(name(n));
      }
      System.out.println(name(0));
      System.out.println(name(26*26*26*26*26*26 - 2));
      System.out.println(name(26*26*26*26*26*26 - 1));
      System.out.println(26*26*26*26*26*26); //308,915,776
   }
}
```

Write a method `public static int nameToInt(String str)`, which returns the n for which `name(n)` is `str` and –1 if `str` is not a valid return value of the `name` method. You may wish to use the method `Math.pow(a, b)`, which returns a raised to the b power, as a double.

Write a methods `public static String next(String str)`, which returns `name(nameToInt(str) + 1)` whenever `nameToInt(str)` is not -1, except `next("ZZZZZZ")` has to be `"AAAAAA"`. For example `next("AAAQGR")` is `"AAAQGS"` and `next("AAAQGZ")` is `"AAAQHA"`. If `str` is not valid, return `null`.

Write a corresponding `previous` method, for example `previous("AAAQGR")` is `"AAAQGQ"`, `previous("AAAQGA")` is `"AAAQFZ"` and `previous("AAAAAA")` is `"ZZZZZZ"`. If `str` is not valid, return `null`.

Add test cases to the main method to show everything works correctly.

Provide a class `Course` with private fields: `static int nextCRN = 10001`, `String title`, `int crn`, `Set<Student> enrollment`--the last 3 are *not* static.
There should be a `Constructor` with one String for the `title`. The constructor sets `crn` to `nextCRN` and then adds 1 to `nextCRN`. The constructor also instantiates `enrollment = new HashSet<>()`.
Write two methods `isUndergrad` and `isGrad` that return `boolean` but both throw `UnsupportedOperationException("No information about level of the course")`

Provide a class `Student` with private fields: `static int nextID = 10001`, `String name`, `int id`, `Set<Course> schedule`, `String major`--the last 4 are *not* static.

There should be a `constructor` with one Strings for the `major`. The constructor sets `id` to `nextID` and then adds 1 to `nextID`. The constructor also instantiates `schedule = new HashSet<>()`. It sets `name` to `Question1.name(id)`.

Write the getter `protected Set<Course> getSchedule()` that returns this Student's schedule.

Write two methods `isUndergrad` and `isGrad` that return `boolean` and both return `false`.

Make a subclass `UndergradCourse` of `Course` and override the methods `isUndergrad` and `isGrad` to return `true` and `false` respectively.

Make a subclass `GraduateCourse` of `Course` and override the methods `isUndergrad` and `isGrad` to return `false` and `true` respectively.

Make a subclass `UndergradStudent` of `Student` and override the method `isUndergrad` to return `true`.

Make a subclass `GraduateStudent` of `Student` and override the method `isGrad` to return `true`. `GraduateStudent` has an additional field `private String undergradMajor;`

The constructors of the subclasses will need to use super(...), where the argument is needed for the constructor of the parent class. The constructor of `GraduateStudent` should also have a second parameter to set the value of `undergradMajor`.

Make a driver class `Question2`, which has an array `public static Course[] courseArray` of 600 courses, where 400 are undergraduate courses 200 are graduate courses, and also has an array `public static Student[] studentArray` of 6000 students, with 5000 undergraduate students and 1000 graduate students. Populate these arrays in a "static block"

```
public class Question2 {
    public static Course[] courseArray;
    public static Student[] studentArray
    static {
    //the code to instantiate and insert values
    //into the two arrays just as if the code were in
    //the main method
    }

    // the main method comes here
}
```

To get the titles and majors, import `java.util.Random`, make an object `rand` of type `Random` and then use `Question1.name(rand.nextInt(308915776))` to create arguments for the constructors. Some titles may repeat but the courses will have distinct `crns`.

To make the next part more fun, randomize your two arrays using

```
ArrayList<Courses> list = new ArrayList<Course>(Arrays.asList(courseArray));
Collections.shuffle(list);
courseArray = list.toArray(courseArray);

// and similarly randomize studentArray
```

For every student, randomly pick four courses from the array of courses (use `rand.nextInt(600)` to pick) and add them to the student's schedule--you need to provide an `addCourse` method to `Student`.
Put a `public void adjustSchedule(){}` method is Student. The empty body means it does nothing.

(i) In `GraduateStudent`, override the `adjustSchedule` method so that it counts how many undergraduate courses were assigned in the student's schedule. Allow a maximum of 1 undergraduate course (use the `isUndergrad()` method). For any other undergraduate courses beyond that limit of 1, remove it from the Set (`schedule.remove`) and keep picking other courses until you find a new graduate course to add.
SUGGESTIONS: This is an example of a simple looking customer specification being non-trivial to implement. Write a helper method that counts how many courses in the schedule are undergraduate and in a WHILE loop (while that number is above 1) search in the set for a course that is undergraduate, remove that course and then loop through picking random courses until you find one that is a graduate course AND when you add it to the schedule is not a course you already have--that is easy since the `add` method of the `Set` interface has the very convenient feature of returning true if the object your are adding is different from any of the ones already in the set and false if it is a duplicate.

A SECOND way would be to copy the schedule into an array of Course and loop through with a counter from 0 to 3 and for any index after you have found one undergrad course, do the remove/insert procedure outlined above.

(ii) Similarly override the `adjustSchedule` method in `UndergraduateStudent` so the student has a maximum of 2 graduate courses.

In Student add a method `public boolean hasCourse(int crn)`, which returns true if the student has that course in their schedule (`Course` will need a getter method for `crn`).

In Course, add a method `public void tallyEnrollment(Student[] allStudents)` that gathers those `Students` in `allStudents` that have this course in their `schedule` (use the `hasCourse`method) and stores it in the `enrollment` field.

Provide a getter method in `Course` for the size of the `enrollment` field.

In Question2, compute the maximum class size, the minimum non-empty class size, and the average class size. Also compute the percentage of graduate students that are taking an undergraduate class. Provide a method `public boolean hasUndergradClass()` in the class Graduate Student to help. In order to call that method when going through the array you will need a "cast"

```
...a counter is already declared
if(studnt.isGrad()) {
   if(((GraduateStudent)studnt).hasUndergradClass()) {
      counter++;
   }
}
```

Write classes `BaseClass`, `Derived1 extends BaseClass`, and `Derived2 extends BaseClass`. All three have a different private String field (`field0`, `field1` and `field2` respectively). Write constructors that see all the fields of the classes. The toString of BaseClass is

```
public String toString() {
   return field0;
}
```

Hence `System.out.println(new BaseClass("Hello World"));` prints `Hello World`. Use super.toString() in the toString method of Derived1 to return field0, a space and then field1. Use super.toString() in the toString method of Derived2 to return field2, a space and then field0.

Write four classes `LevelOne extends LevelTwo, LevelTwo extends LevelThree, LevelThree extends LevelFour and LevelFour`. LevelFour has a private field of type int and a constructor with one int parameter that sets the value of the field.

LevelFour has methods `public double measure()` and `public double value()` that both return the value of the int field.

LevelThree has a field that is a *private* array of BankAccount (given below) and a constructor `LevelThree(double[] balances)` that makes an array of length `balances.length` and fills the array with BankAccounts with those balances. The first line of the constructor calls super(balances.length). Note we cannot check for `balances==null` before this call. Override the `measure`method to return the sum of all the balances in the accounts in the BankAccount array.
Also write a method `public double distance(int i, double mean)`, which returns the Math.abs value of the difference between the balance in the i-th element of the array of BankAccounts and the value of `mean`.

In LevelTwo override the measure method to return the average of the balances of the BankAccounts in the array in LevelThree. You need super.measure(). Return 0 if length() is 0. *LevelTwo needs a constructor that takes an array of double that is passed to the constructor of LevelThree using super(...).*

In LevelOne override the `measure` method to return the largest difference between any balances in all the BankAccounts stored in the LevelThree part and the average calculated by `super.measure()`. *LevelOne needs a constructor that takes an array of double that is passed to the constructor of LevelTwo using super(...).*

```
package assignment06;
/**
```

```
   A bank account has a balance that can be changed by
   deposits and withdrawals.
 */
public class BankAccount {
        private double balance;

        /**
      Constructs a bank account with a zero balance.
         */
        public BankAccount() {
                // double field is automatically intialized to 0.0
        }

        /**
      Constructs a bank account with a given balance.
      @param initialBalance the initial balance
         */
        public BankAccount(double initialBalance) {
                balance = initialBalance;
        }

        /**
      Deposits money into the bank account.
      @param amount the amount to deposit
         */
        public void deposit(double amount) {
                balance += amount;
        }

        /**
      Withdraws money from the bank account.
      @param amount the amount to withdraw
         */
        public void withdraw(double amount) {
                balance -= amount;
        }

        /**
      Gets the current balance of the bank account.
      @return the current balance
         */
        public double getBalance() {
                return balance;
        }
}
```

assignment6

Updated 2 days ago by Matthew Hems and 2 others

followup discussions *for lingering questions and comments*

Resolved

Unresolved

Anonymous 8 days ago

`public static String nameToInt(String str)` should return int, not String

Reply to this followup discussion

Resolved
Unresolved

Anthony Caligure 7 days ago
Is the package supposed to be assignment07 or assignment06?

> **Anonymous** 7 days ago Assignment 7 will apparently share some classes with this one, so assignment07

Reply to this followup discussion

Resolved
Unresolved

Kevin Cheng 7 days ago Hi all,

Please note that the condition to convert 'a' to 'p' in the radix of 26 to 'K' to 'Z' respectively is updated.

Previously:

```
if(chs2[i] >= 'a' && chs2[i] <= 'z')
chs2[i] = (char)('A' + 10 + chs2[i]-'a');
```

Updated:

```
if(chs2[i] >= 'a' && chs2[i] <= 'p')
chs2[i] = (char)('A' + 10 + chs2[i]-'a');
```

Thanks.

Reply to this followup discussion

Resolved
Unresolved

Ryan 7 days ago how do I convert a character of the string back to its letter or number that the integer.to string originally created. I've tried a lot and I can't figure it out.

> **Leslie Lander** 7 days ago Time to attend an office hour. That's why we have them

> **Leslie Lander** 7 days ago Think of a character 'A' through 'Z' as a number 0 through 25.
>
> If I asked you to convert a String "013752" into the number 13752, don't you think you could do it?
>
> It is explained in https://en.wikipedia.org/wiki/Horner%27s_method, where "x" is 26. only with the x to the power 5 at the front, like the first of the Examples on that page.

Ryan 7 days ago Ahh ok ok that helps a lot, thanks!

Paul 7 days ago If 'Z' corresponds with 25 and 'AAAAAA' corresponds with 000000, would 'ZZZZZZ' correspond with 308,915,775?

Matthew Hems 7 days ago correct. 308,915,775 == (26^6) - 1 just as 999999 == (10^6)-1

Reply to this followup discussion

Resolved
Unresolved

John Rocco 6 days ago The name(int n) method allows for negative values, but should we keep this into account when making nameToInt(String str)?

Leslie Lander 6 days ago Good point. I changed the method.

Reply to this followup discussion

Resolved
Unresolved

Anonymous 4 days ago
For the adjustSchedule method, for every undergrad course that we remove, are we supposed to replace it with a grad course? The wording is slightly confusing. Thanks

Anonymous 4 days ago Yes, for GradStudent's adjustSchedule(). Undergraduate students can have at most 2 graduate courses, so their adjustSchedule() should be different.

Anonymous 2 days ago So are we adding grad courses from the courseArray in Question2 that we haven't used yet to replace the undergrad courses?

Matthew Hems 2 days ago yes

Anonymous 2 days ago One more question, I am not understanding how we are supposed to get an element from the set, test if it's an UndergradCourse, and then remove it. Are we iterating through the courseArray from Question2 and checking if the schedule "contains" the object, if so remove it?

Reply to this followup discussion

Resolved
Unresolved

Anonymous 4 days ago
Is "Write classes `BaseClass`, `Derived1 extends BaseClass`, and `Derived1 extends BaseClass`." a typo?

> **Matthew Hems** 4 days ago Yes, its been fixed to "Write classes `BaseClass`, `Derived1 extends BaseClass`, and `Derived2 extends BaseClass`."

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
In LevelOne, what do we do if the length is 0, just return 0 again?

> **Matthew Hems** 2 days ago Returning 0 is fine for 0 length

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
For "Return the largest difference between any balances in all the BankAccounts stored in the LevelTwo part...", is LevelTwo meant to have its own array field, because right now it does not have anything.

> **Matthew Hems** 2 days ago Looks like a typo - I've changed it to say LevelThree. LevelThree is the class that stores the BankAccount array.

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
Is everything in Question2 Class in the scope of static?

> **Kevin Cheng** 2 days ago Question2 class is a driver, containing two static fields, one static initialization block and the main

method. The static initialization block is to initialize the static fields.

Anonymous 3 hours ago So should all of the computations for Question2 (max class size, average class size, etc.) be done in the main method?

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
Is there supposed to be a LevelFive "LevelThree extends LevelFour and LevelFour"?

Matthew Hems 2 days ago No

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
How would we check for balance as being null if we cannot check it before super in LevelThree Class?

Matthew Hems 2 days ago The call to the super constructor must be first. One possible workaround is to use a ternary expression which will allow us to change the argument to the super constructor based on whether balances is null.

```
super(balances == null ? 0
: balances.length);
```

which will call the super constructor with 0 if balances is null and call it with the length of balances if it is not null.

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
In Question2:
When initializing the courseArray and studentArray in the static block above main, we should also instantiate "Random rand =..." in order to assign values for the constructor right?If so, we would also have to instantiate another Random object in the main method in order to add random courses to

each student. Was this implied?

When adding four random courses to each student from main in Question2, some students end up with less than four classes because the addCourse ignores duplicates.
Also, after adding courses to each student, I am assuming that we should also call all of the other methods to update the fields in each course and student. Is this correct?

Leslie Lander 1 day ago You can put one static Random object at the start of the class. For each student, you will have to keep adding courses until the size() of the schedule is 4.

You need to adjust all the courses before you fill out the enrollment of the courses

Reply to this followup discussion

Resolved
Unresolved

Anonymous 2 days ago
How are you guys grading this assignment? I noticed that we aren't told to print anything out for most of the assignment. Are you guys using JUnit and making your own test cases?

Leslie Lander 1 day ago Are you offering something to help?

Matthew Hems 11 hours ago We grade on how complete and correct the code is against what was asked for in the prompt. JUnit may be involved. You are always encouraged to supplement the test code with more than what is asked for in the assignment prompt

Reply to this followup discussion

Resolved
Unresolved

Anonymous 1 day ago
When adjusting the schedules for both Undergrad/Grad students, is it safe to assume that the only courses that will be added to each student's schedule is from the courseArray?
Just checking because if not, there could be an infinite loop when adding/removing courses that aren't present.

Leslie Lander 1 day ago Right you

should be picking new courses from the courseArray that was already build at the start.

Reply to this followup discussion

Resolved
Unresolved

Anonymous 1 day ago
What should

```
public double distance(int i, doub
le mean)
```

return if the i'th index is out of bounds?

Leslie Lander 1 day ago In my code you are in a loop where i is in range: for(int i = 0; i < length(); i++) {

Leslie Lander 1 day ago throw an IllegalArgumentException if i is out of range but the plan is only to call distance from the loop above

Reply to this followup discussion

Resolved
Unresolved

Anonymous 1 day ago
My average class size is 40.0 no matter what, is this correct? I printed out the different course sizes and they're all different but the sum of them is always the same number because every student is in 4 classes, is this what you're looking for?

Leslie Lander 1 day ago Could be: think of how many students you have and how many courses they are in (4 each). That 24000 enrollments in 600 courses. The average can only be 40. I assume the maximum varies.

Anonymous 1 day ago But not every class has 40 students, since it is completely random and students can theoretically all be in only 4 classes, so the average could vary, right?

Anonymous 1 day ago If all the students are in 4 classes, then you have 596 classes that are empty, that you still have to account for in the average. So pretty sure the average will always be the same

Reply to this followup discussion

Resolved

Unresolved

Anonymous 4 hours ago

Wouldn't we want to create a ".removeCourse()" method in the Student class to complete our .adjustSchedule() method in the GraduateStudent class?

Reply to this followup discussion

Resolved

Unresolved

Anonymous 1 hour ago

Is it okay for this assignment to use a break statement within the function `public boolean hasCourse(int crn)` to avoid unnecessary testing once the crn was found or should I just avoid break statements?