Computer Science 145

Final Exam—Spring 2012

Name: ______________________________

Problem	Score	Possible
1		20
2		20
Total		40

This is a closed-book, no-calculator, no-electronic-devices, individual-effort exam. You may reference one page of handwritten notes. All answers should be clearly written. Questions that require code should be written using correct Java syntax. Please do all your work on these pages. Partial credit will be given if work is shown and is partially correct. You may write `SOP` to represent `System.out.println`.

Class	Method/Constructor	Description
`Scanner`	`Scanner(System.in)`	create `Scanner` for parsing `System.in`
	`Scanner(String text)`	create `Scanner` for parsing `text`
	`String next()`	get next delimited word
	`double nextDouble()`	get next delimited `double`
	`boolean nextBoolean()`	get next delimited `boolean`
	`int nextInt()`	get next delimited integer
`String`	`int length()`	get number of characters
	`char charAt(int i)`	get the character at index `i`
	`String toUpperCase()`	get a `String` like this one, but in all-caps
	`String substring(int l)`	get substring from index `l` to `String`'s end
	`String substring(int l, int r)`	get substring from index `l` to right before index `r`
`Math`	`int max(int a, int b)`	get the maximum of `a` and `b`.
	`double pow(double base, double exponent)`	raise `base` to the `exponent` power.
`Random`	`Random()`	create a random number generator.
	`nextInt(int i)`	get a random number between 0 and $i-1$, inclusive.
	`nextDouble()`	get a random number between 0.0 and 1.0.

1. *The Ratrix*

   Wireless communication company SellPhones, Inc., has hired you to implement a monthly billing system. They offer customers two different data plans: Dataholic and Economy. They've got a `Customer` object started with a couple of constants defined:

   ```
   class Customer {
     public static final int DATAHOLIC = ...;
     public static final int ECONOMY = ...;
     ...
   ```

   To complete their monthly billing system, augment the class according to the following tasks:

   (a) Declare two instance variables, one for the customer's selected service plan (either `DATAHOLIC` or `ECONOMY`) and one for the number of megabytes transferred this month (which may have fractional values). Do not assign these variables any values; that is the responsibility of the constructor and other methods that you are not asked to write.

   (b) Customers under the Dataholic service plan get unlimited access to the Internet and other data-based services. They pay a flat $50 a month. Complete the method to return the amount this customer would owe under the Dataholic plan.

   ```
   private double getDataholicBill() {
   ```

(c) Customers under the Economy service plan pay $20 per month and can transfer 200 megabytes of data. For each megabyte over 200, they pay an extra $0.10. Complete the method to return the amount this customer would owe under the Economy plan.

```
private double getEconomyBill() {
```

(d) The previous two methods calculated the customer's bill under the two possible service plans. Now, to complete `getBill`, take into consideration the customer's actual selected service plan. Return the amount due. Use your methods above.

```
public double getBill() {
```

2. *Bored Games*

   Bingo is a zero-strategy game in which a player marks off randomly drawn numbers from a 5-by-5 board. The mechanics of the game are not relevant to this problem. Complete the following problems to create a `BingoBoard` object.

   (a) Start a class `BingoBoard` with one instance variable: a 5-by-5 2-D `int` array representing the board.

   (b) Create a `private` and `static` helper method named `getCell` that takes two parameters:

      i. A `Random` object

      ii. An `int` column number, assumed to be in [0, 4]

      This method returns a semi-random `int` whose value depends on the column number. If column is 0, a number in [1, 15] is returned. If 1, [16, 30]. If 2, [31, 45]. If 3, [46, 60]. If 4, [61, 75]. Recall how `Random.nextInt` works.

(c) Create a `private` and `static` helper method named `getCellUnique` that takes three parameters:

   i. A `Random` object
   ii. An `int` column number, assumed to be in [0, 4]
   iii. A 75-element `boolean` array

   This method returns a semi-random `int` appropriate for the given column (see `getCell`) that also has not been generated on a previous call. One can tell if a number `i` has been previously generated by examining element `i - 1` in the array parameter. If the element is `true`, the number has already been picked. Once you land on an unpicked number, set the element to `true` so a future call to this method will not regenerate it.

(d) Create a constructor that has no parameters. It assigns to the instance variable array a random board. Values in column 0 are all in [1, 15], column 1 in [16, 30], etc. The board contains no repeated numbers. Use your `private` helper methods. (With them, this method can be completed in nine lines of code or so.) The middle cell is called the free space; assign it value 0.

Have a nice summer!