APCS A DS - Finals Practice

The `HomeBaker` class below provides a method for selecting the next *recipe.* By extending this class, different cooking strategies can be modeled.

```
public class HomeBaker
{
    private String name;  // name of this HomeBaker

    /**
     * Creates a HomeBaker with the given name.
     *
     * @param aName    the name of the HomeBaker
     */
    public HomeBaker(String aName)
    {
       name = aName;
    }

    /**
     * Retrieves the name of the HomeBaker.
     *
     * @return  the HomeBaker's name
     */
    public String getName()
    {
       return name;
    }

    /**
     *   This implementation chooses the first valid recipe (based on the contest state).
     *   Override this method in subclasses to define HomeBakers with other cooking strategies.
     *
     *   @param state  the current state of the contest; its current HomeBaker is this HomeBaker.
     *   @return  a string representing the recipe chosen;
     *           "no recipe"  if no valid recipes for the current HomeBaker.
     */
    public String getNextRecipe(Contest state)
    { /*  implementation not shown  */ }
}
```

The method `getNextRecipe()` returns the next *recipe* to be made as a string, using the same format as that used by `makeNextRecipe()` in `ContestRounds`. Depending on how the `getNextRecipe()` method is implemented, a HomeBaker can exhibit different cooking strategies.

(a) Write the complete class declaration for a `RandomHomeBaker` class that is a subclass of `HomeBaker`. The class should have a constructor whose `String` parameter is the RandomHomeBaker's name. It should override the `getNextRecipe()` method to randomly select one of the valid recipes in the given contest state. If there are no valid recipes available for the RandomHomeBaker, the string `"no recipe"` should be returned.

Write the `RandomHomeBaker` class below.

```
public class RandomHomeBaker extends HomeBaker {
    public RandomHomeBaker (String s) {
        super (s);
    }

    @Override
public String getNextRecipe (Contest state) {
  return state.getCurrentRecipes ().get( (int) (Math.random () *
        state.getCurrentRecipes().size () ));

}


}
```

(b) The `ContestRounds` class is used to manage the state of the recipes during a contest. The `ContestRounds` class can be written without knowing details about the contest being enacted.

```
public class ContestRounds
{
    private Contest tent;        // the current state of the contest

    public ContestRounds (Contest initial)
    {
        tent = initial;
    }

  /**
   * Enacts the complete contest round. As long as there is no winner,
   * it alternates among HomeBakers. For each HomeBaker, it allows that HomeBaker to
   * make a recipe.
   *
   * When the contest is over, it should stop making recipes and print either
   * the name of the winner and the word "wins" or, if there is no winner,
   * the message "Contest ends in a draw".
   */
    public void bakeOff()
    {   /* to be implemented in part (b) */ }

  // There may be fields, constructors, and methods that are not shown.
}
```

Write the `ContestRounds` method `bakeOff()`. It should repeatedly determine the current HomeBaker and that HomeBaker's next *recipe*, and make the *recipe*. When the contest is over, it should stop making recipes and print either the name of the winner and the word `"wins"` or the message `"Contest ends in a draw"` if there is no winner. You may assume that the `ContestRounds makeNextRecipe()` method has been implemented so that it will properly handle any *recipe* description returned by the `HomeBaker getNextRecipe()` method, including the string `"no recipe"`.

Use this space for scratch work.

Complete method `bakeOff()` below.

```
/**
 * Runs the contest. As long as there is no winner,
 * it alternates among HomeBakers. For each HomeBaker, it allows that HomeBaker to
 * make a recipe then goes on to the next HomeBaker.
 *
 * When the contest is over, it should stop making cooking recipes and print either
 * the name of the winner and the word "wins" or, if there is no winner,
 * the message "Contest ends in a draw".
 */
public void bakeOff() {
    while (!tent.isShowOver()) {
        tent.makeNextRecipe(tent.getCurrentHomeBaker().
                    getNextRecipe(tent));
    }

    HomeBaker winner = tent.getWinner();
    if (winner == null)
        System.out.println("Contest ends in a draw");
    else
        System.out.println(winner.getName() + " wins");

}
```

Scanned with CamScanner

The `GreatBritishBakeOff` implements a `Contest` and contains the state for a contest that has multiple rounds for multiple HomeBakers. The HomeBakers are kept in a `List`, and the `getCurrentHomeBaker()` method selects the next HomeBaker from the `List` of HomeBakers, making that HomeBaker the current HomeBaker. Suppose that a contest with three HomeBakers is being played out. For example, if initially the `HomeBaker List` might look like:

HomeBaker1	HomeBaker2	HomeBaker3

The first call to `getCurrentHomeBaker()` method returns `HomeBaker1`; however, before returning `HomeBaker1`, it rotates the HomeBakers so that the `HomeBaker List` looks like this:

HomeBaker2	HomeBaker3	HomeBaker1

The next time that `getCurrentHomeBaker()` method is called it would return `HomeBaker2`; however, before returning `HomeBaker2`, it rotates the HomeBakers so that the `HomeBaker List` looks like this:

HomeBaker3	HomeBaker1	HomeBaker2

The contest continues likewise with the HomeBakers rotating within the `HomeBaker List.` A partial implementation of `GreatBritishBakeOff` is shown below and on the next page.

```
public class GreatBritishBakeOff implements Contest
{
  // instance fields to be written in part (c)

   /**
    * Creates a GreatBritishBakeOff object with a list of HomeBakers.
    *
    * @param listOfHomeBakers a list of the contest's HomeBakers
    */
   public GreatBritishBakeOff(List<HomeBaker> listOfHomeBakers)
   { /* to be written in part (c) */ }


  /**
   * Determines whether a winner of the contest exists.
   *
   * @return true if the contest is in an ending state; otherwise,
   *         false
   */
  public boolean isShowOver()
  {/* not shown */}
```

Write the `GreatBritishBakeOff getCurrentHomeBaker()` method below. See the earlier diagrams and explanation for a visual on how the implementation is to be done. The diagrams contest how the current HomeBaker is placed at the end of the list before this method exits.

```
/**
 * Retrieves the HomeBaker who is to make the next recipe, making that HomeBaker the
 * current HomeBaker.
 *
 * @precondition isContestOver() returns false
 *
 * @return the HomeBaker who is to make the next recipe
 */
public HomeBaker getCurrentHomeBaker()
{
~~return bakers.next();~~
~~}~~

q.add(q.remove(0));
return q.get(q.size()-1);

}
```

Scanned with CamScanner

APCS First Semester MC Worksheet

Name ______________________________ Period_____ Date_________

Multiple-choice Questions.

1. What will be the output of the following code fragment?

```
int x = 5;
int y = 8;
int z = 4;
System.out.println("Answer = " + x + y + z/5);
```

concat same prec as add

```
A. A compile error is output.
B. Answer = 580
C. Answer = 13
D. Answer = 3.4
E. Answer = 3
```

2. What will be the output of the following code fragment?

```
int x = 5;
int y = 8;
int z = 4;
System.out.println("Answer = " + (x + y + z) /5.0);
```

```
A. A compile error is output.
B. Answer = 580
C. Answer = 13
D. Answer = 3.4
E. Answer = 3
```

3. What will be the output of the following code fragment?

```
int x = 5;
int y = x + x;
String s = "5";
String t = s + s;
System.out.println( (x + y) + s + t);
```

```
A. 10555
B. 15555
C. 51055
D. 510555
E. 60510
```

8. Look at the following poorly formatted program segment. If `x = 8` and `y = 5` before execution, which of the following represents the correct values of `y`, `z`, `s`, and `t` after execution? An undetermined value is represented with a question mark?

```
if (x == 6)
if (y == 5)
{
   z = 10;
   y = 10;
}
else
{
   s = 20;
   if (y == 5)
         y = 15;
}
else r = 2;

A. y =  5,  z = ?,  r = 2,  s = ?
B. y = 15,  z = ?,  r = ?,  s = 20
C. y =  5,  z = ?,  r = ?,  s = ?
D. y = 15,  z = ?,  r = 2,  s = 20
E. y = 10,  z = ?,  r = ?,  s = ?
```

9. Consider the following program segment:

```
/*
 * @precondition a[0] … a[n-1] is an initialized array of integers
 *                 in which at least one element is negative, and
 *                 0 < n <= a.length
 */
int c = 0;
for (int i = 0 ; i < n ; i++)
{
      if (a[i] <= 0)
      {
            a[c] = a[i];
            c++;
      }
}
```

What is the best postcondition for the segment?

A. `a[0] … a[c]` has been stripped of all positive integers.
B. `a[0] … a[c]` has been stripped of all negative integers.
C. `a[0] … a[c]` has been stripped of all nonnegative integers.
D. `a[0] … a[c]` has been stripped of all non positive integers.
E. None are correct.

APCS First Semester MC Worksheet

10. Consider the following code:

```
/**
 * Removes one character from a String.
 * @return: a string in which the character at index n has
 *          been removed.
 * @precondition:  str is a non-empty string and
 *                 0 <= n < str.length().
 */
public String removeCharacter(String str , int n)
{
        // implementation
}
```

Which of the following are valid implementations?

```
I.   return str.substring(0,n) + str.substring(n-1);
II.  return str.substring(0,n) + str.substring(n);
III. return str.substring(0,n) + str.substring(n+1);
```

A. I only
B. II only
C. III only
D. I and II only
E. II and III only

11. Which of the following correctly constructs a `Date` object?

```
A. Date d = new (2, 13, 1947);
B. Date d;
   d = new (2, 13, 1947);
C. Date d;
   d = Date (2, 13, 1947);
D. Date d = new Date (2, 13, 1947);
E. Date d = Date (2, 13, 1947);
```

12. Which of the following statements will correctly calculate the average of the following integers n1, n2, and n3?

```
A. int average = (n1 + n2 + n3) / 3;
B. double average = n1 + n2 + n3 / 3;
C. double average = (n1 + n2 + n3) / 3.0;
D. double average = (double) ((n1 + n2 + n3) / 3);
E. double average = (n1 + n2 + n3) / 3;
```

13. Which of the following will correctly extract the tens digit of an integer number stored as n?

```
A. (n / 10) % 100
B. (n / 10) % 10
C. (n % 10) /10
D. (n / 100) %10
E. (n % 10) % 10
```

APCS First Semester MC Worksheet

14. A programmer wants to add 1 to a variable called `counter`. Which are valid ways to add 1?

```
I.     counter = counter + 1;
II.    counter += 1;
III.   counter++;
```

A. I only
B. II only
C. III only
D. I and III only
E. I, II, and III

15. Consider three initializations of variables that will hold the summation (`sum`) and the products (`product`) of the int elements in an array.

```
I.   int sum = 0;
  int product = 0;
II.  int sum = 1;
  int product = 1;
III. int sum = 0;
  int product = 1;
```

Assume the loop that calculates the sum and the product work correctly. Which of the above initializations will work correctly?

A. I only
B. II only
C. III only
D. I and II only
E. I and III only

16. Consider the following code segment:

```
Student st1 = new Student("Smith", "May 17, 1994");
Student st2;
st2 = st1;
st1 = new Student ("Patel","January 2, 1995");
```

Which of the following is correct?

(A) This will not compile because `st1` is declared twice
(B) Both `st1` and `st2` refer to the same student with the name of Smith.
(C) Both `st1` and `st2` refer to the same student with the name of Patel.
(D) `st1` and `st2` are two different students, and `st1` is named Smith.
(E) `st1` and `st2` are two different students, and `st1` is named Patel.

17. Which of the following statements are equal to `!a && !b`

```
A.   !a || !b
B.   !(a && b)
C.   !(a || b)
D.   true
E.   false
```

$\bar{a} \cdot \bar{b} = \overline{a+b}$

$\bar{a} + \bar{b} = \overline{ab}$

APCS First Semester MC Worksheet

Questions 18 – 20 rely on the following, incomplete definition of the Book and BookStore classes:

```
public class Book
{
      private double price      // the price of this book
      /**
       * @return the price of this book
       */
      public double getPrice( )
      {  /* implementation not shown */ }
}

public class BookStore
{
      /**
       * @return the sum of the prices of the books in the inventory array
       */
      public double totalPrice(Book[] inventory)
      {  /* implementation not shown */ }
            double sum = 0.0;
            for (int k = 0; k < inventory.length; k++)
            {
                  // MISSING CODE
            }
            return sum;
      }
}
```

18. Consider adding code to the Book class that would permit the price of the book to be set when a variable of type Book was defined. For example:

```
Book b1 = new Book(10.50);
Book b2 = new Book(25.00);
```

Which of the following best describes the code that should be written?

A. A constructor with no arguments
B. A constructor with one argument.
C. A constructor with two arguments
D. A method names setPrice
E. It is not possible to write code that would work as specified.

APCS First Semester MC Worksheet

19. Which of the following code segments could be used to replace `// MISSING CODE` in method `totalPrice` so that it works as specified by its `@return` comment.

```
A. sum += inventory.price[k];
B. sum += inventory.price.getPrice(k0;
C. sum += inventory.Book[k];
D. sum += inventory[k].Book();
E. sum += inventory[k].getPrice();
```

20. Consider adding another method to the `BookStore` class with the following header:

```
public double totalPrice(List<Book> inventory)
```

Which of the following statements about the proposed new method is true?

A. It is an example of inheritance.
B. It is an example of an interface.
C. It is an example of overloading.
D. It is an example of recursion.
E. It is an example of casting.

21. Consider the following code segment:

```
int[ ] x = initializeArrayToRandomInts();
int[ ] y = initializeArrayToRandomInts();
for (int k = 0; k < x.length && k < y.length && x[k] == y[k] ; k++)
{
  {   /* implementation not shown */ }
}
```

Which of the following must be true after executing this code segment?

```
A. k < x.length || k < y.length
B. k < x.length || k < y.length && x[k] == y[k]
C. k < x.length || k < y.length && x[k] != y[k]
D. k < x.length || k < y.length || x[k] == y[k]
E. k < x.length || k < y.length || x[k] != y[k].
```

APCS First Semester MC Worksheet

22. Suppose the following three classes are defined and each has a no-args constructor (a constructor that takes no parameters):

```
public class Vehicle { ... };
public class Car extends Vehicle { ... };
public class Buick extends Car { ...};
```

Which of the following is NOT a legal statement?

```
A. Vehicle vehicle = new Car( );
B. Vehicle vehicle = new Car( );
C. Car vehicle = new Buick( );
D. Car vehicle = new Vehicle( );
E. Buick buick = new Buick( );
```

23. Consider the following method:

```
/**
 *   Counts and prints the number of times a value d is doubled before it is at least as large as target.
 *   @param d        d > 0.0
 *   @param target   target >= 0.0
 */
public void doubleUp (double d, double target)
{
   int count = 0;
   while (d < target)
   {
      count = 0;
      d *= 2;
      count++;
   }
   System.out.println(count);
}
```

Of the following, which best describes the error in `doubleUp`?

A. The value of `target` is not valid.
B. The variable is declared in the wrong place.
C. A `double` cannot be multiplied by an `int`.
D. The counter is incremented in the wrong place.
E. The variable is assigned a value in the wrong place.

24. What is the value of 217_{OCT} as a decimal number?

A. 47_{DEC}
B. 79_{DEC}
C. 133_{DEC}
D. 143_{DEC}
E. 214_{DEC}

APCS First Semester MC Worksheet

Question 25 is based on the `Computable` interface and `LargeInt` class shown below. Many details such as instance variables and implementation are not shown:

```
public interface Computable
{
   /** Returns this object + obj */
   Object add(Object obj;

   /** Returns this object - obj */
   Object subtract(Object obj;

   /** Returns this object * obj */
   Object multiply(Object obj;
}


public class LargeInt complements Comparable, Computable
{
   /** Converts n to a LargeInt object */
   public LargeInt(int n)
   {  < implementation is not shown > };

   /** Returns this object as a String*/
   public String toString( )
   {  < implementation is not shown > };

   /** Returns this object + obj */
   public Object add(Object obj
   {  < implementation is not shown > };

   /** Returns this object - obj */
   public Object subtract(Object obj
   {  < implementation is not shown > };


   /** Returns this object * obj */
   public Object multiply(Object obj;
   {  < implementation is not shown > };

   /**   Returns -1 if this LargeInt is less than obj;
    *           1 if this LargeInt is greater than obj;
    *           0 if this LargeInt is equal to obj
    */
   public int compareTo(Object obj;
   {  < implementation is not shown > };

   /**   Returns true if this LargeInt is less than obj; otherwise,
    *           false
    */
   public boolean equals(Object obj;
   {  < implementation is not shown > };
}
```

APCS First Semester MC Worksheet

25. Of the following pairs of methods, which should be coded and tested as soon as possible to facilitate testing and debugging the other methods?

    A. The constructor and `add` method
    B. The constructor and `compareTo` method
    C. The constructor and `toString` method
    D. The `toString`, `equals`, and `compareTo` methods
    E. The `toString` and one of the `add`, `subtract`, or `multiply` methods

26. What is the value of **AE**$_{\text{HEX}}$ as a decimal number?

    A. 31_{DEC}
    B. 32_{DEC}
    C. 174_{DEC}
    D. $175_{\text{DEC.}}$
    E. 256_{DEC}