AP CS Unit 4: Classes and Objects Programs

1. Copy the Coin class and complete the second constructor. Write a second class, named CoinRunner, that has a main method and does the following:

- Create four Coin objects. Give one Coin a value of 5 and another a value of 3. The other two Coins should have random values.
- Display the value of the two Coins with random values.
- Call the getValue method for each object and display the sum of the values.

```
public class Coin {
    private int value;

    public Coin( int v ){
     value = v;
    }

    public Coin(){
     // assign value a random integer
between 1 and 10
    }

    public int getValue(){
     return value;
    }
}
```

2. Copy the Bucket class. Make sure it compiles (but you won't be able to run it because it does not have a main method).

Add a second class (named BucketRunner) that does contain a main method. The main method should do the following:

- Create 2 Bucket objects that can each hold 10 gallons.
- Add a random amount to each bucket (use Math.random). The amounts should be decimals between 5 (inclusive) and 12 (exclusive).
- Display how full each bucket is (e.g. 74% and 81%)
- Empty each bucket.
- Call the percentFull method again for each object and display the returned values (they should both be zero).

```
public class Bucket {
    private double amt;
    private double capacity;

    public Bucket( double c ) {
     capacity = c;   // gallons
     amt = 0;
    }

    public void emptyBucket() {
     amt = 0;
    }

    public void add ( double stuff ) {
     amt += stuff;
     if ( amt > capacity)
          amt = capacity;
    }

    public double percentFull( ){
     return 100.0 * amt / capacity;
    }
}
```

3. Copy the Walker method and complete the walk method. Make sure it compiles. Add a second class (named RunWalk) and enter this code:

```
public class RunWalk {
    public static void main(String[] args) {
     Walker w1 = new Walker();
     w1.walk( 5 );  // moves 1 - 5 (random)
     w1.walk( -4 );   // should not move
     w1.walk( 8 );  // moves 1 - 8 (random)
     w1.walk( 1 );  // moves 1 foot
     int loc = w1.getX();
     System.out.println( "w1 is now at " + loc );
    }
}
```

Run it and make sure it works.

4. Add a third class that contains a main method. In this main method create another Walker and keep calling the walk method (with an argument of 10) until its x value is 100 or greater. After it reaches 100, display its x value and the number of times the walk method was called.

```
public class Walker {
    private int x;  // location in feet

    public Walker() {
     x = 0;
    }

    public void walk( int max ){
     /* If max is less than one this method
does nothing. If max is positive, then this
method generates a random number
between one and max (inclusive) and
increases x by that amount. */
    }

    public int getX(){
     return x;
    }
}
```

5. Complete the Star class.

Second, write a runner class that creates three Star objects and calls their display methods. Then it calculates and displays the distance between the three stars. If the stars create an isosceles or equilateral triangle, it displays a message to that effect (though this is extremely unlikely to actually happen). If they do not form an isosceles or equilateral triangle, then the program determines and displays which two stars are closest to each other. Here's a sample output:

```
Star 1 coordinates: 15, 20, 18
Star 2 coordinates: 7, 14, 15
Star 3 coordinates: 14, 18, 20
The distance from star 1 to star 2 is
10.44030650891055
The distance from star 2 to star 3 is
9.486832980505138
The distance from star 3 to star 1 is 3.0
Star 3 and Star 1 are the closest
```

```
public class Star{
    private int x, y, z;

    public Star(){
     assign the instance variables random values
between 0 and 20
    }

    public double distance( int x1, int y1, int z1){
     return the distance from (x, y, z)
                              to (x1, y1, z1)
}

    public void display(){
     System.out.println(
          "coordinates: " + x + ", " + y + ", " + z );
    }

    Add three accessor methods to the Star class.

}
```

6. Complete the removeDups method in the ProblemX class. Then write a runner class and paste this code in the main method to test your solution.

```
ProblemX p1 = new ProblemX( "Eels" );
String s = p1.removeDups();
System.out.println( s );          // els
s = p1.toString();
System.out.println( s );          // eels

ProblemX p2 = new ProblemX( "AaAaAaAh!!!" );
s = p2.removeDups();
System.out.println( s );          // ah!
s = p2.toString();
System.out.println( s );          // aaaaaaah!!!

ProblemX p3 = new ProblemX( "" );
s = p3.removeDups();
System.out.println( s );          // empty string

ProblemX p4 = new ProblemX( "12221122334231" );
s = p4.removeDups();
System.out.println( s );          // 121234231
s = p4.toString();
System.out.println( s );          // 12221122334231
```

```
public class ProblemX{
    private String str;

    public ProblemX( String s ){
        str = s.toLowerCase();
    }

    public String removeDups(){
        Returns a string where any
duplicate adjacent characters have
been removed.  The instance variable is
not changed.
    }

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

7. Complete the addCommas method in the StrNumber class. Then write a runner class and paste this code in the main method to test your solution.

```
StrNumber sn1 = new StrNumber( "7" );
System.out.println( sn1.addCommas() );       // 7
StrNumber sn2 = new StrNumber( "23" );
System.out.println( sn2.addCommas() );       // 23
StrNumber sn3 = new StrNumber( "405" );
System.out.println( sn3.addCommas() );       // 405
StrNumber sn4 = new StrNumber( "6183" );
System.out.println( sn4.addCommas() );       // 6,183

StrNumber sn5 = new StrNumber( "12345678" );
System.out.println( sn5.addCommas() );       // 12,345,678

StrNumber sn6 = new StrNumber( "71399372947382" );
System.out.println( sn6.addCommas() );
                                  // 71,399,372,947,382

StrNumber sn7 = new
                    StrNumber( "commas are important" );
System.out.println( sn7.addCommas() );
                                  // co,mma,s a,re ,imp,ort,ant
```

```
public class StrNumber{
    private String nums;

    public StrNumber( String s ){
        nums = s;
    }

    public String addCommas(){
        Returns a string that has a
comma placed after every third
character starting from the end of
the string.  The returned string
will never start with a comma.
Check the test code for examples
on what it should return.
    }
}
```

8. Review the comments below and then complete the isPalindrome method in the Pal class.

Write a runner class and paste this code in the main method to test your solution.

```
Pal p1 = new Pal( "radar" );
System.out.println( p1.toString() );
System.out.println( p1.isPalindrome() );

Pal p2 = new Pal( "radars" );
System.out.println( p2.toString() );
System.out.println( p2.isPalindrome() );
Pal p3 = new Pal( "Amore, Roma" );
System.out.println( p3.toString() );
System.out.println( p3.isPalindrome() );
Pal p4 = new Pal( "race car?" );
System.out.println( p4.toString() );
System.out.println( p4.isPalindrome() );
```

It should display:

```
original: radar, clean: RADAR
true
original: radars, clean: RADARS
false
original: Amore, Roma, clean:
AMOREROMA
true
original: race car?, clean: RACECAR
true
```

```
public class Pal{
    private String orig;
    private String clean;

    public Pal( String s ){
        orig = s;
        clean = "";
        s = s.toUpperCase();
        for ( int k = 0; k < s.length(); k++ ) {
            char ch = s.charAt(k);
            if ( ch >= 65 && ch <= 90 )
                clean = clean + ch;
        }
    }

    public boolean isPalindrome(){
        Returns true if this is a palindrome;
otherwise it returns false
    }

    public String toString(){
        return "original: " + orig +
                ", clean: " + clean;
    }
}
```

Comments.

- Read the opening paragraph of this Wikipedia article on Palindromes. http://en.wikipedia.org/wiki/Palindrome
- Without going into detail, the letter A is represented by the number 65, B is represented by 66 and so on. Lower-case letters start at 97 and go up to 122. The primitive data type char can store exactly one character. Java allows you to compare two chars by using > and <. The compiler will NOT allow you to compare two strings using > and/or <.
- You will not be tested on the char data type but it is useful and you should be able to figure out what is going on in the Pal constructor.