PROBLEM 1 : (*Loop de Loop* (8 pts))

Consider the following code.

```
   String [] names = {"Tsu Chi", "Fa La", "Do Re Me", "Sue Rod"};

   for (int k=0; k<names.length; k++ )
   {
      String temp = names[k];
      names[k] = temp.substring(temp.indexOf(" ")+1);
   }

   for (String str: names)
   {
      System.out.println(str);
   }
}
```

1. Give the output for this code.

2. Give a meaningful loop invariant for the first for loop.

3. Suppose the array *names* is instead initialized to

   ```
   String [] names = {"Duke"};
   ```

   Give the output if the code above is run with this value for *names*.

PROBLEM 2 : (*Basketball country* (15 pts))

Consider the following two classes *BasketballPlayer* and *DukeBasketballPlayer* and note the word **extends** in the second class.

```
public class BasketballPlayer {

   private String myName;   // name of player
   private double myHeight; // height in inches
   private int myStamina;   // low number is low stamina

   // constructor
   public BasketballPlayer(String name, double height)
   {
      myName = name;
      myHeight = height;
      myStamina = 10;
   }

   public String getName()     // return name of player
   {
      return myName;
   }

   public double getHeight()   // return height of player
   {
      return myHeight;
   }

   public int getStamina()     // return stamina of player
   {
      return myStamina;
   }

   // update stamina by adding num to it
   public void updateStamina(int num)
   {
      myStamina+= num;
   }

   public void workout()
   {
      System.out.println(getName() + " working out ...");
      updateStamina(-2);
   }
   public void rest()
   {
      System.out.println("resting ...");
      updateStamina(4);
   }
}

public class DukeBasketballPlayer extends BasketballPlayer{

   private String myNickName;  // players nickname

   // constructor
   public DukeBasketballPlayer(String name, double height, String nick)
```

```
   {
      super(name, height);
      myNickName = nick;
      updateStamina(10);
   }

   public String getName()  // return name of player
   {
      return "NCAA champ: " + super.getName();
   }

   public void workout()
   {
      updateStamina(-1);
      System.out.println(getName() + " super workout ...");
   }

   public void praise()
   {
      System.out.println("Great Game " + myNickName);
   }
}
```

PROBLEM

Consider the following code segments. There are no syntax errors such as missing semicolon, but there may be errors related to how inheritance should work. For each code segment, if there is no error, then give the output. If there is an error then explain which line gives the error and why.

1. ```
   BasketballPlayer gordon = new BasketballPlayer("Hayward", 81.0);
   gordon.workout();
   gordon.rest();
   System.out.println(gordon.getName() + " " + gordon.getStamina());
   ```

2. ```
   DukeBasketballPlayer brian = new DukeBasketballPlayer("Zoubek", 85.0,"Zoob");
   brian.workout();
   brian.rest();
   brian.praise();
   System.out.println(brian.getName() + " " + brian.getStamina());
   ```

3. ```
   BasketballPlayer jon = new DukeBasketballPlayer("Scheyer", 77.0, "Mr. Basketball");
   jon.workout();
   jon.rest();
   System.out.println(jon.getName() + " " + jon.getStamina());
   jon.praise();
   ```

4. ```
   DukeBasketballPlayer kyle = new BasketballPlayer("Singler", 80.0);
   kyle.workout();
   kyle.rest();
   System.out.println(kyle.getName() + " " + kyle.getStamina());
   kyle.praise();
   ```

5.
```
ArrayList<BasketballPlayer> players = new ArrayList<BasketballPlayer>();
BasketballPlayer avery = new BasketballPlayer("Jukes",80.0);
DukeBasketballPlayer miles = new DukeBasketballPlayer("Plumlee", 82.00, "Plumlee brother");
players.add(avery);
players.add(miles);
for (BasketballPlayer player: players)
{
   System.out.println(player.getName());
}
```

PROBLEM 3 : (*Naming Names* (12 pts))

Consider the Student class.

```
public class Student {
  private String myName;  // name of Student
  private String myYear;  // class of student such as first-year, sophomore, etc.

  public Student(String name, String year) {
     myName = name;
     myYear = year;
  }
  public String getName() {
     return myName;
  }
  public String getYear() {
     return myYear;
  }
}
```

Write the method `uniqueNames` whose header is given below. This method takes as input an ArrayList of type Student and a year indicator and returns a TreeSet of the unique names of all students whose year matches the year indicator.

For example, if an ArrayList of Student types called Dook contained the following five persons (indicating their name and year):

```
Ed first-year, Ma sophomore, Jo first-year, Qiang first-year, Jo first-year
```

then the call `uniqueNames(Dook, "first-year")` would return a TreeSet containing the strings: "Ed", "Jo" and "Qiang".

```
public TreeSet<String> uniqueNames(ArrayList<Student> people, String year)
{


}
```

PROBLEM 4 : (*Reflections in the Warped Mirror* (11 pts))

Here is the code for Negative, to make the Negative of a Pixmap image.

```
public class Negative extends Command
{
    public static final int MAX_COLOR_LEVEL = 255;

    public Negative ()
    {
        super("Reverse Colors");
    }

    public void execute (Pixmap target)
    {
        // loop over each color in pixmap
        Dimension bounds = target.getSize();
        for (int x = 0; x < bounds.width; x++)
        {
            for (int y = 0; y < bounds.height; y++)
            {
                // invert
                Color old = target.getColor(x, y);
                target.setColor(x, y,
                    new Color(MAX_COLOR_LEVEL - old.getRed(),
                              MAX_COLOR_LEVEL - old.getGreen(),
                              MAX_COLOR_LEVEL - old.getBlue()));
            }
        }
    }
}
```

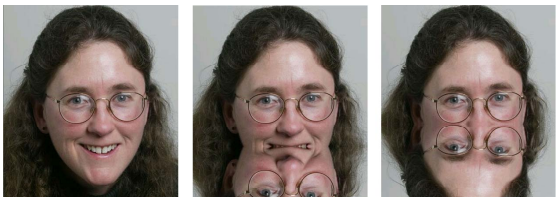
You will write code for another Pixmap problem on the next page.

Write the `execute` method of `Reflect` whose header is given below. This method takes a color Pixmap image and modifies the picture to show the reflection about a horizontal line that is randomly chosen in the bottom half of the picture.

As an example, suppose the image height is 442 pixels. Then a random line must be chosen in the bottom half of the pixmap, between half the height, 221, and 442. Suppose 402 is randomly chosen. Then the pixels from 403 to 442 must be replaced with the reflection values (403 with 401, 404 with 400, 405 with 399, etc.)

For an image example consider the picture shown on the left below. The second picture is the result of the random horizontal line that was chosen happened to be in the mouth area. The reflection is from the mouth down. Starting with the leftmost picture again, the third picture is the result of the random horizontal line that was chosen happened to be near the bottom of the nose.

```
public class Reflect extends Command
```



```
{
    // code not shown that is not needed

    public void execute (Pixmap target)
    {
        Dimension bounds = target.getSize();   // size of pixmap
        // TODO: complete method below

    }
}
```

PROBLEM 5 : (*Feel good: give to charity (26 points)*)

The following problem stores information about donors and how much they give to charity organizations.

First consider the Donor class that keeps track of a name and the amount of money this donor has given to a particular charity. If the donor has given several times to this charity, then there is only one Donor with that name, and myDonations will be the sum of all those donations.

```
public class Donor {

   String myName;       // name of donor
   double myDonations;  // total amount of donations

   // constructor
   public Donor (String name, double amount)
   {
      myName = name;
      myDonations = amount;
   }

   // return name of donor
   public String getName()
   {
      return myName;
   }

   // return donor's total donations
   public double getDonations()
   {
```

```
      return myDonations;
   }

   // increase current donation by "amount"
   public void makeDonation(double amount)
   {
      myDonations += amount;
   }
}
```

Next consider the Charity class on the next page that keeps track of all the donors for one charity and how much money each donor has given to the particular charity. Note that there are unique entries for each donor. If a donor gives additional money to the charity, it is added to the current donor record.

```
public class Charity {

   String myName; // name of charity
   ArrayList<Donor> myDonors;   // Donors for this charity

   // constructor
   public Charity(....)  { }         // code not shown

   // returns the name of the charity
   public String getCharityName() {
      return myName;
   }

   // given a donor name, return amount donated to this charity
    public double totalAmountFromDonor(String donorName) { } // code not shown

   // return an ArrayList of all the donors
   public ArrayList<Donor> getDonors() {
      return myDonors;
   }

   // add donor if new donor to the end of the ArrayList, otherwise combine donation
   //    with current donor info
   public void addDonor(String name, double amount) { } // code not shown

   // print the name of the charity and a list of donors and their amounts
   public void printCharityInfo()
   {
      System.out.println(``Charity is `` + myName + ``:'');
      for (Donor d: myDonors)
      {
          System.out.println(``\t'' + d.getName() + `` $'' +
            d.getDonations());
      }
   }
}
```

PART A. Consider the following code. What is the output?

```
Charity museum = new Charity(``Nasher'');
museum.addDonor(``Mel Gates'', 150.00);
museum.addDonor(``Jo Tan'', 500.00);
museum.addDonor(``Mel Gates'', 2000.00);
museum.addDonor(``Bob Jo Yu'', 300.00);
museum.printCharityInfo();
```

List the output here:

PART B. Write the *totalAmountFromDonor* method in the *Charity* class. Remember that there is one unique entry of each donor for this charity. This method is given the name of a donor and it returns the amount of money this donor has given to this charity.

```
// given a name, return the amount they donated
public double totalAmountFromDonor(String donorName)
{
```

PART C. Now consider the Organization class shown below. An Organization can keep track of information for lots of charities with its ArrayList of type Charity.

```
public class Organizations {

    private ArrayList<Charity> myOrganizations; // data for all charities

    // all items not shown

    // constructor
    public Organizations() {
       myOrganizations = new ArrayList<Charity>();
    }

    // return the total donations a person gave to all charities
    public double totalDonations(String person)  { } // code not shown

    // return the name of the donor who gave the most money to charities
    public String DonorWithLargestDonations()  { } // code not shown

    // return an ArrayList of the unique donor names
   public ArrayList<String> getAllDonors()   {  }  // code not shown
}
```

Complete the Organization method *DonorWithLargestDonations()*. This method returns the name of the donor whose total donations to all the charities is the largest. Assume there is only one such donor.

For example, assume Organizations contains the following information (assume this data has already been read into the private data of the class).

```
Charity is Museum Life and Science, Donors are:
        Fred Smith $150.0
        Gosling $300.0
        Sa Lou Duke $700.0
Charity is Duke Nasher, Donors are:
        Mary Jo Tan $1500.0
        Mel Gates $3000.0
        Gosling $100.0
Charity is UNC Public Radio, Donors are:
        Mary Jo Tan $50.0
        Chao $1100.0
        Sa Lou Duke $250.0
Charity is United Way, Donors are:
        Gosling $600.0
        Mel Gates $2000.0
        Chao $1200.0
```

Then the call *DonorWithLargestDonations()* would return *Mel Gates*, who gave a total of $5000, more than any other donor.

Complete *DonorWithLargestDonations()* below.

NOTE that you can call other methods whose code are not shown such as *totalDonations* and *getAllDonors*. You do not have to write code for these methods to use them.

```
// return the name of the donor who gave the most money to charities
public String DonorWithLargestDonations()
{









}
```

CLASS methods

```
public class String {
    public int length ()  // length of string
    // Returns a substring of this string that
    // begins at the specified beginIndex and
    // extends to the character at index
    // endIndex - 1.
    public String substring (int beginIndex,
                             int endIndex)
    // Returns a substring of this string that
    // begins at the specified beginIndex
    public String substring (int beginIndex)
    // Returns position of the first
    // occurrence  of str, -1 if not found
    public int indexOf (String str)
    // Returns the position of the first
    // occurrence of str from index start
```

```
    public int indexOf (String str, int start)
    // returns character at position index
    public char charAt(int index)
    // returns true if str has the exact
    // same characters in the same order
    public boolean equals(String str)
    // splits into Array based on delim
    String [] split(String delim)
}

public class Collections {
   // Sorts the specified list according
   // to the order induced by thecomparator
   public static void sort(List list,
                        Comparator c)
}

public class Integer {
    // Returns the argument as a signed integer.
    public int parseInt(String s)
}

public class Pixmap {
    public Dimension getSize()
    // Get the pixel at (row,col)
    public Color getColor (int row, int col)
    // Set the pixel at (row,col)
    public void setColor (int row, int col, Color value)
}

pubclic class Dimension {
    public int width
    public int height
}

public class Color {
    // Creates a color with the specified red,
    // green, and blue values in the range
    // (0 - 255)
    public Color(int r,int g,int b)
    // Returns color components
    public int getRed()
    public int getGreen()
    public int getBlue()
}

public class Random {
    // Create a new random number generator
    public Random()
    // Returns a pseudorandom, uniformly
    // distributed value in [0,n)
    public int nextInt(int n)
}
```

```
public class ArrayList {
    // Constructs an empty list
    public ArrayList ()
    // Returns the number of elements
    public int size ()
    // Returns element at position index
    public Object get (int index)
    // Replaces the item at position index
    // with element.
    public Object set (int index, Object element)
    // Appends specified element
    public boolean add (Object o)
}

public class Scanner {
    // Create Scanner that reads data from a file.
    public Scanner (File file)
    // Create Scanner that reads data from a string.
    public Scanner (String str)
    // Change delimiters used to separate items
    public void useDelimiter (String characters)
    // Check if more items are available
    public boolean hasNext ()
    // Get next delimited item as a string
    public String next ()
    // Get next line as a string
    public String nextLine ()
    // Get next delimited item as an integer value
    public int nextInt ()
    // Get next delimited item as a Double value
    public double nextDouble ()
}
```

```
public class TreeSet
{
   // creates an empty TreeSet
   public TreeSet()

   // adds object e to the TreeSet
   boolean add(Object e)

   // adds all elements from a collection to this set
   boolean addAll(Collection c)

   // Retains only the elements in this set that are contained
   // in the specified collection - returns true if set changed
   boolean retainAll(Collection c)

   // Removes from this set all of its elements that are contained
   // in the specified collection
   boolean removeAll(Collection c)

   // removes all objects from the TreeSet
   void clear()

   // returns true if e is in the set, othewise returns false
   boolean contains(Object e)

  // returns true if set is empty, otherwise returns false
  boolean isempty()

  // returns an Iterator for the set
  Iterator<Object> iterator()

  // removes the object e from the set
  boolean remove(Object e)

  // returns the number of elements in the set
  int size()

}
```

To find the size of a built-in array, use length.