Week 5 Homework Q24

Telmen Enkhbold

San Fransico Bay University

CE480 - Java and Internet Application

Dr. Chang, Henry

10/12/2023

# Author Note

# The Question

1. Implement the class Ghost
   * Attributes
     + name (String data type)
     + dynamic number of eye sizes (double data type)
     + dynamic number of nose lengths (int data type)
     + dynamic number of mouth sizes (float data type)

Note:

* + - The class [**Vector**](https://hc.labnet.sfbu.edu/~henry/sfbu/course/introjava/java_class/slide/vector.html)can be used to implement dynamic objects.
    - [Sample code](https://hc.labnet.sfbu.edu/~henry/sfbu/course/introjava/java_class/slide/classStruct.htm#Class%20with%20dynamic%20numerical%20array%20attribute)
  + Member functions
    - Helping function
    - Manager functions
      * Constructor
    - Access functions
      * get
      * set
      * Predicate
        + isNormal()
        + isNoMouth()
        + isBlind()  
          Note: A ghost is blind if

it has eyes, but the sizes of the eyes are 0, or

it does not have eye.

* + - * + isNoNose()
        + isNoEye()
    - Implementor
      * void convertToNormal()  
        A normal ghost has 2 eyes, 1 nose, 1 mouth
      * void kill()
      * void addOneEye(int size)
      * void removeFirstMouth()
      * void removeLastNose()
      * void enlargeFirstMouth(float added\_size)
      * String toString()
  + Test your class by
    - Step 1: Create 3 objects
      * casper
        + name="Casper"
        + eye: {2.3 , 2.3}
        + nose: {2}
        + mose: {2.1}
      * mike
        + name="Mike Wazowski"
        + eye: {5.2}
        + nose: No nose
        + mose: { 6.3 }
      * james
        + name="James Sullivan"
        + eye: {3.3, 3.3}
        + nose: {2}
        + mose: {5.3}
    - Step 2: Convert the object **mike** into a normal one
    - Step 3: Check whether the objects are normal after the conversion.
    - Step 4: Kill Casper
    - Step 5: Enlarge **mike**'s moth by 2.1
    - Step 6: Remove **james**'s nose
    - Step 7: Add one eye to **mike**
    - Step 8: Remove **mike**'s mouth
    - Step 9: Display the content of the object **mike**
    - Step 10: Display the content of the object **james**
  + References
    - [Class with dynamic numerical array attribute](https://hc.labnet.sfbu.edu/~henry/sfbu/course/introjava/java_class/slide/classStruct.htm#Class%20with%20dynamic%20numerical%20array%20attribute)
    - [Class with fixed numerical array attribute](https://hc.labnet.sfbu.edu/~henry/sfbu/course/introjava/java_class/slide/classStruct.htm#Class%20with%20fixed%20numerical%20array%20attribute)

A screenshot of a computer program

Description automatically generated

The Source Code

import java.util.Vector;

public class Ghost {

  private String name;

  private Vector<Double> eyeSizes;

  private Vector<Integer> noseLengths;

  private Vector<Float> mouthSizes;

  // Constructor

  public Ghost(

    String name,

    Vector<Double> eyeSizes,

    Vector<Integer> noseLengths,

    Vector<Float> mouthSizes

  ) {

    this.name = name;

    this.eyeSizes = eyeSizes;

    this.noseLengths = noseLengths;

    this.mouthSizes = mouthSizes;

  }

  // Access functions for name

  public String getName() {

    return name;

  }

  public void setName(String name) {

    this.name = name;

  }

  // Access functions for eyeSizes

  public Vector<Double> getEyeSizes() {

    return eyeSizes;

  }

  public void setEyeSizes(Vector<Double> eyeSizes) {

    this.eyeSizes = eyeSizes;

  }

  // Access functions for noseLengths

  public Vector<Integer> getNoseLengths() {

    return noseLengths;

  }

  public void setNoseLengths(Vector<Integer> noseLengths) {

    this.noseLengths = noseLengths;

  }

  // Access functions for mouthSizes

  public Vector<Float> getMouthSizes() {

    return mouthSizes;

  }

  public void setMouthSizes(Vector<Float> mouthSizes) {

    this.mouthSizes = mouthSizes;

  }

  // Predicate: isNormal() - Check if the ghost is normal

  public boolean isNormal() {

    return (

      eyeSizes.size() == 2 && noseLengths.size() == 1 && mouthSizes.size() == 1

    );

  }

  // Predicate: isNoMouth() - Check if the ghost has no mouth

  public boolean isNoMouth() {

    return mouthSizes.isEmpty();

  }

  // Predicate: isBlind() - Check if the ghost is blind

  public boolean isBlind() {

    return (

      eyeSizes.isEmpty() || (eyeSizes.size() > 0 && eyeSizes.get(0) == 0)

    );

  }

  // Predicate: isNoNose() - Check if the ghost has no nose

  public boolean isNoNose() {

    return noseLengths.isEmpty();

  }

  // Predicate: isNoEye() - Check if the ghost has no eyes

  public boolean isNoEye() {

    return eyeSizes.isEmpty();

  }

  // Implementor: convertToNormal() - Make the ghost normal

  public void convertToNormal() {

    eyeSizes.clear();

    eyeSizes.add(2.0);

    eyeSizes.add(2.0);

    noseLengths.clear();

    noseLengths.add(1);

    mouthSizes.clear();

    mouthSizes.add(1.0f);

  }

  // Implementor: kill() - Remove all features of the ghost

  public void kill() {

    eyeSizes.clear();

    noseLengths.clear();

    mouthSizes.clear();

  }

  // Implementor: addOneEye(int size) - Add an eye with the given size

  public void addOneEye(double size) {

    eyeSizes.add(size);

  }

  // Implementor: removeFirstMouth() - Remove the first mouth

  public void removeFirstMouth() {

    if (!mouthSizes.isEmpty()) {

      mouthSizes.remove(0);

    }

  }

  // Implementor: removeLastNose() - Remove the last nose

  public void removeLastNose() {

    if (!noseLengths.isEmpty()) {

      noseLengths.remove(noseLengths.size() - 1);

    }

  }

  // Implementor: enlargeFirstMouth(float addedSize) - Enlarge the first mouth by the specified size

  public void enlargeFirstMouth(float addedSize) {

    if (!mouthSizes.isEmpty()) {

      float currentSize = mouthSizes.get(0);

      mouthSizes.set(0, currentSize + addedSize);

    }

  }

  @Override

  public String toString() {

    return (

      "Name: " +

      name +

      "\nEye Sizes: " +

      eyeSizes +

      "\nNose Lengths: " +

      noseLengths +

      "\nMouth Sizes: " +

      mouthSizes

    );

  }

  public static void main(String[] args) {

    // Step 1: Create 3 objects

    Vector<Double> casperEyeSizes = new Vector<>();

    casperEyeSizes.add(2.3);

    casperEyeSizes.add(2.3);

    Vector<Integer> casperNoseLengths = new Vector<>();

    casperNoseLengths.add(2);

    Vector<Float> casperMouthSizes = new Vector<>();

    casperMouthSizes.add(2.1f);

    Ghost casper = new Ghost(

      "Casper",

      casperEyeSizes,

      casperNoseLengths,

      casperMouthSizes

    );

    Vector<Double> mikeEyeSizes = new Vector<>();

    mikeEyeSizes.add(5.2);

    Vector<Integer> mikeNoseLengths = new Vector<>();

    Vector<Float> mikeMouthSizes = new Vector<>();

    mikeMouthSizes.add(6.3f);

    Ghost mike = new Ghost(

      "Mike Wazowski",

      mikeEyeSizes,

      mikeNoseLengths,

      mikeMouthSizes

    );

    Vector<Double> jamesEyeSizes = new Vector<>();

    jamesEyeSizes.add(3.3);

    jamesEyeSizes.add(3.3);

    Vector<Integer> jamesNoseLengths = new Vector<>();

    jamesNoseLengths.add(2);

    Vector<Float> jamesMouthSizes = new Vector<>();

    jamesMouthSizes.add(5.3f);

    Ghost james = new Ghost(

      "James Sullivan",

      jamesEyeSizes,

      jamesNoseLengths,

      jamesMouthSizes

    );

    // Step 2: Convert the object mike into a normal one

    mike.convertToNormal();

    // Step 3: Check whether the objects are normal after the conversion

    System.out.println(

      "Is Casper normal after conversion? " + casper.isNormal()

    );

    System.out.println("Is Mike normal after conversion? " + mike.isNormal());

    System.out.println("Is James normal after conversion? " + james.isNormal());

    // Step 4: Kill Casper

    casper.kill();

    // Step 5: Enlarge Mike's mouth by 2.1

    mike.enlargeFirstMouth(2.1f);

    // Step 6: Remove James's nose

    james.removeLastNose();

    // Step 7: Add one eye to Mike

    mike.addOneEye(3.0);

    // Step 8: Remove Mike's mouth

    mike.removeFirstMouth();

    // Step 9: Display the content of the object Mike

    System.out.println("Mike's Ghost Object:\n" + mike);

    // Step 10: Display the content of the object James

    System.out.println("James's Ghost Object:\n" + james);

  }

}