Questions 1-10 refer to the following program.
3 points each question.

```
package edu.nyu.cs.finale;

/* assume that the following imports all exist and do not contain errors */
import edu.nyu.cs.finale.Concept
import edu.nyu.cs.finale.Being;
import edu.nyu.cs.finale.Nothingness;

public class ConceptualUnderstanding {
	private String accumulatedWisdom[] = new String[];
	private int wisdomCounter = 0;
	private static Concept[] concepts;

	public static void main(String[] args) {
		this.concepts = {
			new Being(),
			new Nothingness()
		};
		for (int i=0; i<concepts.length; i++) {
			String details = concepts[i].getSummary();
			try {
				if (this.comprehend(details)) {
					System.out.println("Got it!");
				}
				else {
					System.out.println("Skipping...");
					continue;
				}
			}
			catch (LackOfConceptualClarityException e) {
				System.out.println("Sorry, we're lost.");
			}
			catch (GeneralizedExistentialException e) {
				System.out.println("We must carry on.");
			}

		}
	}

	public static comprehend(String details) {
	    if (details.indexOf("monism") > 0 || details.length > 500) {
		throw new LackOfConceptualClarityException();
		return false;
	    }
	    else {
		this.accumulatedWisdom[this.wisdomCounter] = details;
		this.wisdomCounter++;
		return true;
	    }
	}
}
```

Questions 1-5
Identify 5 unambiguous syntactic or logic errors in this code and clearly explain the mistake being made and how to fix it. Draw arrows with numbers indicating each error in the code.

//Error #1

//Error #2

//Error #3

//Error #4

//Error #5

Questions 6-10 require short answers.
Please keep answers short and to the point.

//Question #6
Explain which part of this code exhibits polymorphism, and how so.

//Question #7
Can you say for sure whether the Concept class has any child classes?
Explain why or why not.

//Question #8
Does this code suggest that the Concept class’s getSummary() method may throw a LackOfClarityException? Explain why or why not.

//Question #9
Explain abstraction and indicate where it is exhibited in this code, if at all.

//Question #10
Explain how ArrayLists differ from arrays and how they can be used to improve certain parts of this code.

Questions 11-15 refer to the following code.
5 points each question.

```
/**
 * Class that represents an x,y coordinate in 2D space
*/
class Point {
	int x = 0;
	int y = 0;

	public Point(int x, int y) {
		this.x = x;
		this.y = y;
	}
}
```

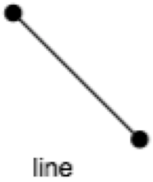
//Question #11

Write a main method that instantiates two Point objects representing two different coordinates in 2D space.

//Question #12

According to how object-oriented programming ideology is implemented in Java, should the two instance variables in the Point class be declared as public, protected, or private? Justify your answer.

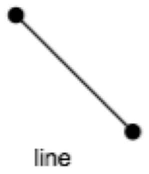
//Question #13

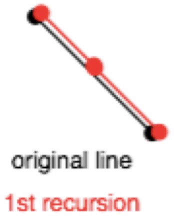


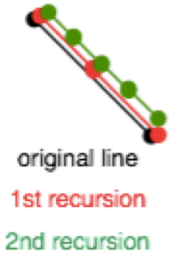
Define a class that represents a Line in 2D space. Use instances of the Point class from the previous questions to represent any Line object’s beginning and ending coordinates. Include a main method that instantiates one of these Line objects with specific beginning and ending Points.

//Question #14

Write a new recursive version of the Line class mentioned in the previous question. Here’s how this one should differ: when any Line object is instantiated, the code should automatically cause two more Line objects to be instantiated: one representing a line from the beginning Point of the original line to a new Point representing the midpoint of that original Line, and the other representing a line from that midpoint Point to the end Point of the original Line. This should recursively call itself for 3 recursive levels.







Continue answer to Question 14 here

//Question 15
How many Line objects total will be instantiated when the code created in the previous question executes.

Questions 16-25 are multiple choice.
3 points each question.

//Question 16
Each element in a two-dimensional array must have the same number of sub-elements.

a. True
b. False

//Question 17
A class can implement more than one interface.

a. True
b. False

//Question 18
Constructors in a class definition… (select all that apply):

a. …must accept no parameters
b. …must have no return value
c. …must be overloaded at least once
d. …must not be declared as public or private

//Question 19
The keyword “super” is a reference to a class’s… (select all that apply):

a. no-args constructor
b. parent class
c. instantiated object
d. package
e. access modifier

//Question 20
To adhere to the concept of abstraction, where possible, data fields in a Java program should be made….

a. public
b. static
c. private
d. void
e. -None of the above-

//Question 21
Assume that the following code is encapsulated within a larger project. What is output at the end of execution of the main method? Assume the Cat class exists, and this code does not produce an error.

```
public static void main(String[] args) {
      Cat cuddles = new Cat();
      cuddles.setName(“Cuddles”);
      doggify(cuddles, “Killer”);
      System.out.println(cuddles.getName());
}


public static void doggify(Cat c, String name) {
         c.setName(name);
         c.setDisinterestedInHumans(false);
}
```

a. “Cuddles”
b. “Killer”
c. null
d. –None of the above-

//Question 22
What is output at the end of execution of the main method? Assume this code does not produce an error.

```
public static void main(String[] args) {
     String cuddles = "Cuddles";
     doggify(cuddles, "Killer");
     System.out.println(cuddles);
}


public static void doggify(String s, String name) {
        s = name;
}
```

a. "Cuddles"
b. "Killer"
c. null
d. –None of the above-

//Question 23
The == operator performs a comparison….

a. by "reference"
b. by value

//Question 24
The special keyword, *this*, when used within a static method of a class, refers to…

- a. the current class
- b. the current object
- c. the parent object
- *d. –All of the above-*
- *e. –None of the above-*

//Question 25 (7 points)

Write a class that has the attributes and methods represented in the following UML diagram. Getter and setter methods for all private properties have not been included in the UML diagram, but must be included in the code. Include a constructor that assigns values to all instance variables.

Exam
-questions : String[]
-difficultyLevel : int
-author : String
+askNextQuestion() : void

Calling the askNextQuestion() method should output a single question, starting from the first question in the questions[] array, and continuing through the remaining elements in questions[] array with each subsequent call, until there are no more questions left. Once there are no more questions left, calling this method should simply output a message that there are no more questions.

Show how to instantiate an Exam object, assign it some sample questions and other instance variables, and have it output those questions.

Continue answer to Question 26 here

//Question 26 (8 points)

Create a ComputerScienceCourse class and a Student class, using the following outlines. Anything not included or specified in the outline left up to your discretion.

ComputerScienceCourse	**Student**
title : String	name : String
description : String	email : String
days : String array (a String for each day, such as	year : int (use 1 for freshman, 2 for
time : String (such as “9:30-10:45”)	grade: String
students : Student array (an array of 50 Student objects)	assignGrade(grade : String)
registerStudent(student :	

a. Include a main() method that creates a ComputerScienceCourse object representing this course.
b. Create a Student object representing yourself, and one other Student object representing someone else.
c. Register both Student objects for this course by calling the registerStudent() method and passing it the Student objects.
d. The registerStudent() method should add these two Students to the first two spots in the ComputerScienceCourse object’s students array.
e. Assign the Student objects whatever grade you think you deserve by calling the Student objects’ assignGrade() method. This should update the Student objects’ grade property.

Continue answer to Question 27 here

This page is extra scrap paper

This page is extra scrap paper