CSC330 Final Review

1. a) For this question, you are to write the following Robot class. This class has two private instance variables: speed, which is a double, and mass, which is also a double. It has two constructors: a default constructor that sets speed and mass to 0; and a constructor that has ***two parameters***, one for each member variable. It also has accessor and mutator methods for each variable.

Write the Robot class on the sheet provided.

Public class Robot{

Private double speed;

Private double mass;

Robot(){

This.speed = 0;

This.mass = 0;

}

Robot(double s, double m){

Speed = s;

Mass = s;

}

Public Double getSpeed(){

Return speed;

}

Public Double getMass(){

Return mass;

}

Public Void setMass(double m){  
 mass = m;

}

Public Void setSpeed(double s){  
 speed = s;

}

}

1. For this question, you are to write a subclass of the Robot class from question 1a. The RoboTank class *is-a* Robot with some extra characteristics. A RoboTank has three instance variables: treadLength, which is a double, hasSensor which is a boolean, and minSensorDistance, which is a double. It has two constructors: a default constructor that sets treadLength to 0, hasSensor to false, and minSensorDistance to 1; and a parameterized constructor that has ***five parameters*** which initializes all member variables, including those from Robot. It has accessor and mutator methods for its instance variables, and a method called sensesSomething, defined as follows:

/\*\* Returns a boolean value if this RoboTank senses something

\* near to it. It returns true if it has a sensor and distance

\* is less than or equal to minSensorDistance; false otherwise.

\*

\* @return if this RoboTank has sensed something near to it.

\*/

public boolean sensesSomething(double distance)

Complete the RoboTank class on the exam paper provided.

Public class RoboTank extends Robot{

Private double treadLength;

Private Boolean hasSensor;

Private double minSensorDistance;

RobotTank(){

Super(m,s); Super is optional

treadLength = 0;

hasSensor = False;

minSensorDistance = 1;

}

RoboTank(double m, double s, double tl, Boolean hs, double msd){  
 super(m,s);

treadLength = tl;

hasSensor = hs;

minSensorDistance = msd;

}

Public Double getTreadLength(){

Return treadLength;

}

Public Boolean getHasSensor(){

Return hasSensor;

}

Public Double getMinSensorDistance(){

Return minSensorDistance;

}

Public Double setTreadLegth(double tl){

treadLength = tl;

}

Public Boolean setHasSensor(Boolean hs){

hasSensor = hs;

}

Public Double setMinSensorDistance(double msd){  
 minSensorDistance = msd;

}

Public Boolean sensesSomething(double distance){

If(hasSensor == True && distance < getMinSensorDistance()){

Return True;

}

Else{

Return False;

}

}

}

1. This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.
2. A number group represents a group of integers defined in some way. It could be empty, or it could contain one or more integers.

Write an interface named NumberGroup that represents a group of integers. The interface should have a single contains method that determines if a given integer is in the group. For example, if group1 is of type NumberGroup, and it contains only the two numbers -5

and 3, then group1.contains(-5)would return true, and group1.contains(2)

would return false.

Write the complete NumberGroup interface. It must have exactly one method.

Public interface NumberGroup{

Public Boolean contains(int num);

}

1. A *range* represents a number group that contains all (and only) the integers between a minimum value and a maximum value, inclusive.

Write the Range class, which is a NumberGroup. The Range class represents the group of int values that range from a given minimum value up through a given maximum value, inclusive. For example, the declaration

NumberGroup range1 = Range(-3, 2);

represents the group of integer values -3, -2, -1, 0, 1, 2

Write the complete Range class. Include all necessary instance variables and methods as well as a constructor that takes two int parameters. The first parameter represents the minimum value, and the second parameter represents the maximum value of the range. You may assume that the minimum is less than or equal to the maximum.

Public class Range implements NumberGroup{

Private int min;

Private int max;

Public Range(int min, int max){

This.min = min;

This.max = max;

}

Public Boolean contains(int integer){

If(integer >= min && integer <= max){

Return true;

}

Else{

Return false;

}

}

Public int getMin(){

Return min;

}

Public int getMax(){

Return max;

}

}

1. The MultipleGroups class (not shown) represents a collection of NumberGroup objects and is a NumberGroup. The MultipleGroups class stores the number groups in the instance variable groupList (shown below), which is initialized in the constructor.

private List<NumberGroup> groupList;

Write the MultipleGroups method contains. The method takes an integer and returns true if and only if the integer is contained in one or more of the number groups in groupList.

For example, supposed mulitple1 has been declared as an instance of MultipleGroups and consists of the three ranges created by the calls new Range(5, 8), new Range(10, 12), and new Range(1, 6). The following table shows the results of several calls to contains.

Table

Description automatically generated

Complete method contains on the paper provided. Here is the method header:

/\*\* Returns true if at least one of the number groups in

\* this multiple group contains num; false otherwise.

\*/

public boolean contains(int num)

for(NumberGroup group : groupList){

if(group.getMin() < num && group.getMax() > num){

return True;

}

}

Return False;

3) Consider the hierarchy of classes shown in the following diagram.

Diagram

Description automatically generated

Note that a Cat “is-a” Pet, a Dog “is-a” Pet, and a LoudDog “is-a” Dog.

The class Pet is specified as an abstract class as shown in the following declaration. Each Pet has a name that is specified when it is constructed.

public abstract class Pet

{

private String myName;

public Pet(String name)

{ myName = name; }

public String getName()

{ return myName; }

public abstract String speak();

}

The subclass Dog has the partial class declaration shown below.

public class Dog extends Pet

{

public Dog(String name)

{ /\* implemenetation not shown \*/ }

public String speak()

{ /\* implementation not shown \*/ }

}

1. Given the class hierarchy shown above, write a complete class declaration for the class Cat, including implementations of its constructor and method(s). The Cat method speak returns “meow” when it is invoked.

Public class Cat extends Pet{

Public Cat(String name){

Super(name);

}

Public String speak(){

Return “meow”;

}

}

1. Assume that class Dog has been declared as shown at the beginning of the question. If the String *dog-sound* is returned by the Dog method speak, then the LoudDog method speak returns a String containing *dog-sound* repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class LoudDog, including implementations of its constructor and method(s).

Public class LoudDog extends Dog{

Public String speak(){

Return super.speak() + super.speak();

}

1. Consider the following partial declaration of the class Kennel.

Text

Description automatically generated

Complete method allSpeak on the sheets provided. Here is the method header:

// postcondition: for each Pet in the kennel, its name followed

// by the result of a call to its speak method

// has been printed, one line per Pet.

public void allSpeak(){

for(Pet pet : petList){

System.out.print(pet.Speak());

}

}

4) Choose two of the software development life cycle models we covered in class. Define them and discuss the advantages and disadvantages of those models on the exam paper provided.

Big Bang Model – No process, Choas, Little planning, Requirements implemented on the fly.

Advantages – Very simple, little planning, easy to manage, very few resources, flexible, good for newcomers.

Disadvantages- Very high risk, not a good model for complex objects, poor model for long projects. Can turn out very expensive if requirements are misunderstood.

Spiral Model – Allows incremental releases of production. 4 phases.Identification – gather requirements, Design – conceptual design of software, constructing – Proof of concept is developed to get customer feedback. Evaluation – End of first iteration, customer evaluates and provides feedback.

Advantages – Changing requirements can be accommodated and more accurate. Users can see system early, can be divided into smaller parts and risky parts can be developed earlier which helps with risk management.

Disadvantages – management and process is more complex. Spiral may go on indefinitely, large number number of stages require lots of documentation. Not suitable for small or low risk projects.

5) This question deals with writing your own Exception, and filling in some blanks of a small GUI application, whose purpose is to check for a valid SSN. The following is the code for the SSNChecker class. Answer (a) through (j) on the exam paper provided.

Text

Description automatically generated

When completed, the program should look like:

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Write a SocSecException class. SocSecException *is-a* Exception that calls its parent class’ constructor in its own constructor to set the value of its message.

Public class SocSecException{

Pubic String message(String msg){

Return super(msg);

}

}

Answer the following questions on the paper provided:

//b) declare a Label variable called inputSSNLabel,

// and set its value to "Enter SSN: "

Label inputSSNLabel = new Label(“Enter SSN: “);

//c) declare a Button named checkButton, and set its text to //"Check SSN"

Button checkButton = new Button(“Check SSN”);

//d) if the length of the SSN is not equal to 11,

// throw a new SocSecException with the message

// "SSN length is not 11."

If(inputSSNLabel.length == 11){

Throw new SocSecException(“SSN length is not 11.”);

}

//e) set the text of messageLabel to "Valid SSN!"

messageLabel.setText( “Valid SSN!”);

//f) print the stack trace of sse

Sse.printStackTrace();

//g) set the messageLabel text to the message of sse

messageLabel.setText(sse.getMessage());

//h) declare and instantiate an HBox named hbox1 with a

// spacing of 10, which holds inputSSNLabel, inputSSNField // and checkButton, in that order

HBox hbox1 = new Hbox(10,inputSSNLabelm inputSSNField, checkButton);

//i) declare and instantiate a VBox named vbox with a

// spacing of 10, which holds hbox1 and hbox2, in that

// order

VBox vbox = new VBox(10,hbox1,hbox2);

//j) set the title of the primaryStage to "SSN Checker"

primaryStage.setTitle(“SSN Checker”);

6) UML Diagram

Inheritance, abstract classes, interfaces.

Abstract classes are italicized…use / before abstract classes and methods.

Only list data and behavior that is unique to the class, which includes OVERRIDDEN

PARENT METHODS.

+ for public, - for private, # for protected

Regular inheritance is a connected line from parent to child with an open arrow head.

Inheritance from an abstract class is a dotted line with an open arrow head.

Implementing an interface is also a dotted line with an open arrow head.

If you need to draw an interface, put <<interface>> above the interface name.