Q1 Notes for exam

0 Points

1) You are *not* allowed use Eclipse for any part of this exam

2) There are 4 allowed Resources (and nothing more):

(i) OO Design Principles Handout

https://rhit-csse.github.io/csse220/Docs/Handouts/Basic_OO_Principles_for_CSSE220.pdf

(ii) UML Cheatsheet

https://rhit-csse.github.io/csse220/Docs/Handouts/UML_Cheatsheet.pdf

(iii) Box & Pointer Cheatsheet

https://rhit-csse.github.io/csse220/Docs/Handouts/drawing_box_and_pointer.pdf

(iv) A single double-sided 8.5 x 11" paper cheatsheet that you made for yourself

3) Academic honesty is taken seriously, dishonesty can result in a zero on this exam or an F in the course

4) In several places below, you will be asked to provide a UML diagram (or a recursive trace). The best option to give us a UML diagram below is to make it on plantUML using the link plantuml.com/plantuml and then saving the resulting png and uploading the png file for your submission to the question.

---

csse.github.io/csse220/Docs/Handouts/drawing_box_and_pointer.pdf

(iv) A single double-sided 8.5 x 11" paper cheatsheet that you made for yourself

3) Academic honesty is taken seriously, dishonesty can result in a zero on this exam or an F in the course

4) In several places below, you will be asked to provide a UML diagram (or a recursive trace). The best option to give us a UML diagram below is to make it on plantUML using the link plantuml.com/plantuml and then saving the resulting png and uploading the png file for your submission to the question.

Q2 Small design problem

5 Points

You are given a problem statement below. You will then be asked some questions regarding this problem statement.

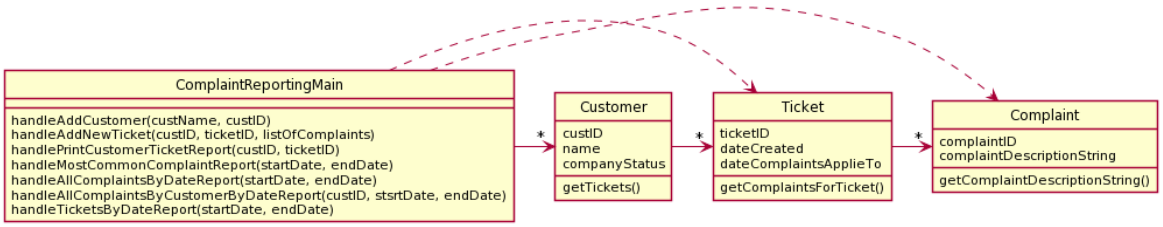
Several companies are tasked with tracking a complaint system for their services. We would like to create a generic complaint system that most companies can use. For such a system, we want to track individual customers, including their names, customer IDs, and their company status, which is something each company can use as a string in whatever way they wish. We also want to track "tickets." When a customer calls to complain about a specific issue, a ticket is created. Each ticket will have an ID, the date the ticket was created and the date that all the complaints of the ticket applies to. Note that if a customer wishes to complain about separate occasions, whether they have different dates, or the separate complaints are not related, a separate ticket must be created for each occasion. Each ticket may have several complaints that relate to the same issue. Note also that ticket IDs are only unique for each customer, so to get a specific ticket, you need the customer ID and ticket ID to identify a unique ticket. Lastly, we want to track the various complaints that exist in the system. A complaint just needs an ID

and a long description string to describe the specific complaint.

For the company, we want various ways of reporting on these complaints. Of course, we want to be able to print a specific ticket report, which contains each complaint in detail for a specific customer. But we would also like things like the following:

- Given a date range, print the most common complaint in that date range
- Given a date range, print all the complaints made in that date range across all customers
- Given a customer ID and a date range, print all the complaints made by that customer in that date range
- Given a date range, print out all ticket reports that were filed in that date range

Given the following UML diagram, there is a problem.



Q2.1 Coupling or Cohesion

1 Point

Would you consider this problem to be related more to Coupling or Cohesion (Hint: consider the design principle that is violated here)?

- (●) Coupling
- ( ) Cohesion

Q2.2 Design fix

Q2.2 Design fix

4 Points

Now, provide a UML diagram to propose a change that fixes the problem you identified in the previous part. To do this, go to plantuml.com/plantuml and create a UML diagram for this question (or you may draw it out by hand and submit a scan). When you are finished, save the image and upload the saved image below. To help get you started, you can click the following link or paste it into your browser address bar. The link will take you to the existing UML diagram (from above) on PlantUML. Once there, you can make changes as necessary rather than starting from scratch.
http://www.plantuml.com/plantuml/uml/bPBVJZ8n4CNl-nGRhlY-e8z0G8Y95w0HNg3PZjZGFvlER0mnlhlJBTFLc8tnffnsUdvpEkmAIlVglLDujg7JlVi6wUSje6UeOk9R3ZjUDUygrSTZ8ErCPiUANf8yvR5NdjOD5azRF-DT_RmThJj6MXUIek6XcrMpY8zljG-ABQdn3k15tV9y8CXZbr4OewFGMZY69wxrSJP89WiQZZdnjPtbrpwgqSoYljRHYlSe96PWoJDuauDKKNP-pxZfduHWpQJozZwb0lb1hDGSIpUogju1jiXtP60wKR3WLvbHrG4zyRZVBPlP_LaUWH40G00

Please submit your file below

No files uploaded

Q3 Big design question

8 Points

Given the problem statement below, answer the following questions.

You are asked to program an interesting and simple "chaser" game. It is "simple" in the sense that while multiple players may play, they all see the same screen and the playing surface does not move during the entire game, and the background is simply a solid color. Each player may choose to be either a Dog or a Cat, and the player may choose what color his/her animal will be. A player moves around the screen (and thus has an x, y position as well as an x, y velocity; the velocity is only non-zero for any player when the

player is actively pressing and holding a movement direction). The Dogs and Cats are supposed to chase a rabbit on screen and the first animal/character to "catch" the rabbit by colliding with it scores 100 points and the rabbit is "respawned" and the game continues until the clock runs out. The player with the highest score when the game time runs out is the winner. The game time is displayed at the upper *left* hand corner of the screen.

The player movement is a little different for this game. The players will use the four directional arrows on their keyboards. However, there is a timer running in the background (displayed at the top *right* corner of the screen). This timer goes off every 30 seconds and affects the different animals as follows:

- For a dog, when the timer goes off, all arrow key directions are randomized. This means that all four directions are still available, but the up arrow key could move the player in ANY of the 4 directions when pressed. The same goes for all 4 arrow keys.
- For a cat, when the timer goes off, one of the four directions, chosen at random, will be "turned off" in the sense that for the next 30 seconds, that direction arrow does not work. We want a design that captures all these rules that will allow for the above game to be implemented.

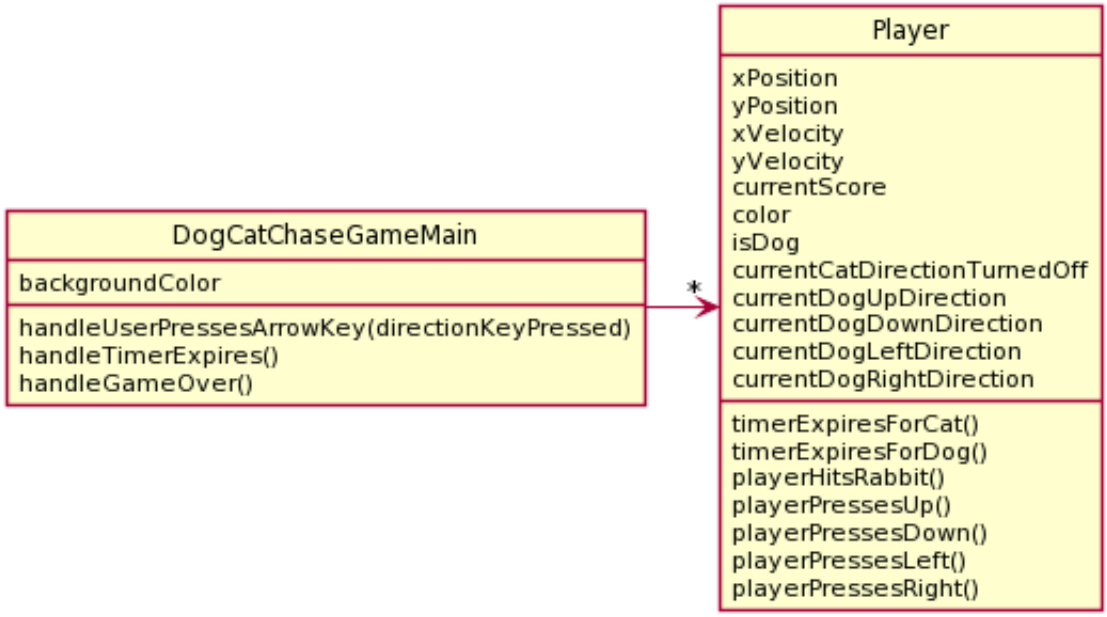
Note that each animal/character is independent of the others, meaning that at any one time, one cat may have the up arrow key turned off, while another cat for another player may currently have the left arrow key turned off. Each character chooses at random without the knowledge of any other character of the same shape.

For this problem, and for simplicity, you can ignore a lot of the extra things, i.e., you can assume the timer(s) is/are implemented elsewhere and that the animation is handled in some other component class not shown, which would display each object at its current position and redraw every 10 milliseconds. The rabbit is also not a part of the designs below.

Q3.1 Solution A

2 Points

The following is one design for the problem statement given above. (*Note, the variable named "currentDogUpDirection" is a variable that has 4 possible values. Each value is the ACTUAL direction the player moves when pressing the up arrow key. For example, if this value is "left", then when the user presses the up arrow key, the player goes left. This applies to the other directional arrow keys as well. Likewise, for cats, the field named currentCatDirectionTurnedOff is the direction that the player currently turned off in the sense that when the player presses that direction, nothing changes.*)



The design above violates one or more of the OO Design Principles. Note, there may be a number principles that are violated, we are looking for the worst one(s).

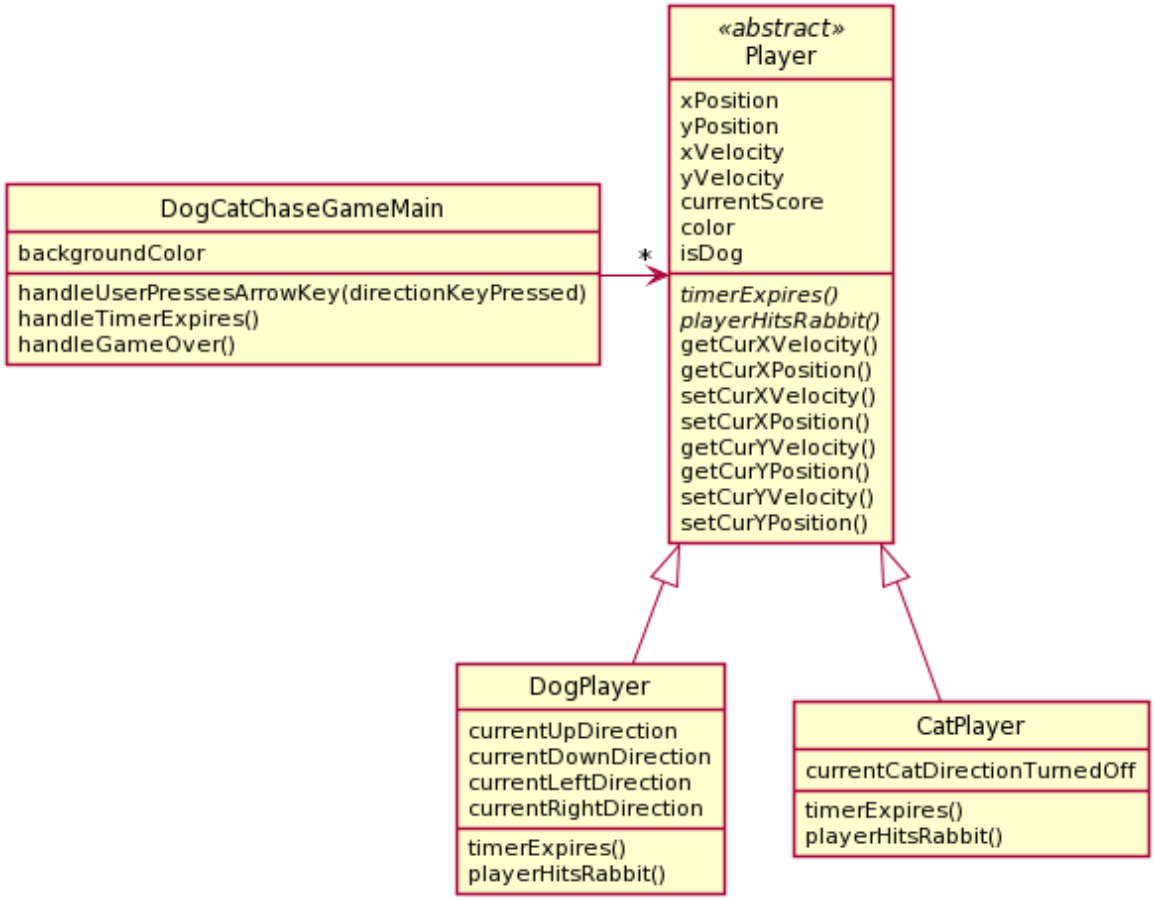
What is/are the principle(s) violated?

Please explain your answer below:

Q3.2 Solution B

2 Points

The following is one design for the problem statement given above.



The design above violates one or more of the OO Design Principles. Note, there may be a number principles that are violated, we are looking for the worst one(s).

What is/are the principle(s) violated?

Please explain your answer below:

Q3.3 Your solution

4 Points

Please provide a UML diagram of a design that fixes the problems of the previous solutions and does not violate any of the OO Design Principles. To do so, you may go to plantuml.com/plantuml, or you may draw it by hand. In either case, please upload the file for your submission.

If you choose to use PlantUML, below are the links to Solution A and Solution B to help get you started.

Solution A:
http://www.plantuml.com/plantuml/uml/RP7TRi8m38NlynGLLp1fhp0nmNwaRG8nsBr
ZeSmnZTqFFM4q4lxx3bvcCw6C9rAi6hplbAG3Upt53BAkzrLGz9O6xB6rf2rbjhDZZth_

Solution B:
http://www.plantuml.com/plantuml/uml/ZPB1Ri8m38RlUOgA4vlKDq0LKvasQPj03APr
lgo0LjjlaboY8WhdDmNzJg556v90ecDeibnnj9y2GBU9FQ18r8fDfcQ2kJnZQtA3RIf3ai
r0uFFJmMcNu1Ed1eBSnoMGC6ev5CY54gZgAW4VkP9NqK5lNvjF-
0t2hDjQZF9rKXWk5tPH64EhdLvBeIZPSx1FpNoKMaFDL-
ranpcIIQNJG3ZYlyz0KwucquGhqSkihnjrOzdLLFPtMfqyivT2_LJghnJl_BYH-
_HnExCptu2clkmRdEDXVqQaQzmNH6OidNHXmwAoqgD91Er-
j_ITxQln2Fri8e6DmFskqGNVTXLORVaUVkyaq-1fEwBVy1

Please upload your file here:

No files uploaded

Q4 Recursive trace

0 Points

This problem is worth **10 points**, but will be submitted on paper. You will be provided sheets of paper and you should do the following problem on that paper. Make sure to put your name on the pages and turn it in before you leave.

Given the recursive code below, we will ask you to provide both the **output** and the **recursive trace** of the method call.

```
3  public class RecursiveTrace {
4
5      public static void main(String[] args) {
6          int[] a = {1, 2, 3};
7          int[] b = {3, 2, 1};
8          System.out.println(mystery(a, b, 0));
9      }
10
11
12     public static String mystery(int[] a, int[] b, int curInd) {
13         if(curInd == a.length)
14             return "";
15
16         int aVal = a[curInd];
17         int bVal = b[curInd];
18
19         if(aVal == bVal) {
20             return mystery(a, b, curInd+1);
21         } else if(aVal < bVal) {
22             String cur = "" + (bVal - aVal); //converts to String
23             return cur + mystery(a, b, curInd+1);
24         } else {
25             String cur = "" + (aVal + bVal);
26             return cur + mystery(a, b, curInd+1);
27         }
28     }
29 }
```

Q5 Inheritance and Polymorphism

19 Points

You will be given a few classes below that utilize inheritance. You will then be

You will be given a few classes below that utilize inheritance. You will then be asked a series of questions regarding these classes and tracing the code polymorphically. (These classes will be repeated for each part for reference.)

```
3  public abstract class Emotion {
4      public abstract void cry();
5      public abstract void laugh(int a);
6
7      public void smile() {
8          System.out.println("Turn that frown...");
9      }
10
11     public void yell() {
12         System.out.println("YAAAH!");
13     }
14 }
```

```
3  public class Happy extends Emotion {
4
5      public void cry() {
6          System.out.println("Finally... Matrix 4!");
7      }
8
9      public void laugh(int a) {
10         System.out.println("They made Fast and Furious " + a);
11         this.smile();
12     }
13
14     public void smile() {
15         System.out.println("Ted Lasso is my hero");
16     }
17 }
```

```
3  public class Sad extends Happy {
4
5      public void cry() {
6          super.cry();
7          System.out.println("WHY end like that Game of Thrones?!?");
8      }
9
10     public void smile() {
11         System.out.println("I had such unmet hope for the new Bill and Ted");
12     }
13
14     public void yell() {
15         System.out.println("Not the Red Wedding!");
16     }
17
18     public void pout() {
19         super.cry();
```

```
20          System.out.println("Another year to wait for season 3...");
21      }
22
23      public void complain() {
24          System.out.println("WHY cancel Firefly?");
25      }
26  }
```

```
 3  import java.util.ArrayList;
 4
 5  public class Feelings {
 6      private Emotion emo;
 7      private ArrayList<Happy> haps;
 8
 9      public Feelings(Sad s) {
10          this.emo = s;
11          this.haps = new ArrayList<Happy>();
12      }
13  }
```

Q5.1 UML

4 Points

Given the classes above, please give a UML Class Diagram that correctly represents the classes provided.

The classes are again:

```
 3  public abstract class Emotion {
 4      public abstract void cry();
 5      public abstract void laugh(int a);
 6
 7      public void smile() {
 8          System.out.println("Turn that frown...");
 9      }
10
11      public void yell() {
12          System.out.println("YAAAH!");
13      }
14  }
```

```java
public class Happy extends Emotion {

    public void cry() {
        System.out.println("Finally... Matrix 4!");
    }

    public void laugh(int a) {
        System.out.println("They made Fast and Furious " + a);
        this.smile();
    }

    public void smile() {
        System.out.println("Ted Lasso is my hero");
    }
}
```

```java
public class Sad extends Happy {

    public void cry() {
        super.cry();
        System.out.println("WHY end like that Game of Thrones?!?");
    }

    public void smile() {
        System.out.println("I had such unmet hope for the new Bill and Ted");
    }

    public void yell() {
        System.out.println("Not the Red Wedding!");
    }

    public void pout() {
        super.cry();
        System.out.println("Another year to wait for season 3...");
    }

    public void complain() {
        System.out.println("WHY cancel Firefly?");
    }
}
```

```java
import java.util.ArrayList;

public class Feelings {
    private Emotion emo;
    private ArrayList<Happy> haps;

    public Feelings(Sad s) {
        this.emo = s;
        this.haps = new ArrayList<Happy>();
    }
```

```
13  }
```

Please use plantUML (or equivalent) to render your UML diagram and submit it below. You do not need to worry about drawing the UML to represent the ArrayList, just the classes given.

No files uploaded

Q5.2 Polymorphic Tracing Part 1 (Simple)

7 Points

Given the declarations below, you will be asked to trace the various pieces of code. For each portion of code, you are asked to type out the output precisely as it would be printed on the console (you may NOT run any of this code in Eclipse, you MUST trace this by hand).

The classes are again:

```
3  public abstract class Emotion {
4      public abstract void cry();
5      public abstract void laugh(int a);
6
7      public void smile() {
8          System.out.println("Turn that frown...");
9      }
10
11     public void yell() {
12         System.out.println("YAAAH!");
13     }
14 }
```

```
3  public class Happy extends Emotion {
4
5      public void cry() {
6          System.out.println("Finally... Matrix 4!");
7      }
8
9      public void laugh(int a) {
10         System.out.println("They made Fast and Furious " + a);
11         this.smile();
12     }
13
```

```
    public void smile() {
        System.out.println("Ted Lasso is my hero");
    }
}
```

```
public class Sad extends Happy {

    public void cry() {
        super.cry();
        System.out.println("WHY end like that Game of Thrones?!?");
    }

    public void smile() {
        System.out.println("I had such unmet hope for the new Bill and Ted");
    }

    public void yell() {
        System.out.println("Not the Red Wedding!");
    }

    public void pout() {
        super.cry();
        System.out.println("Another year to wait for season 3...");
    }

    public void complain() {
        System.out.println("WHY cancel Firefly?");
    }
}
```

```
import java.util.ArrayList;

public class Feelings {
    private Emotion emo;
    private ArrayList<Happy> haps;

    public Feelings(Sad s) {
        this.emo = s;
        this.haps = new ArrayList<Happy>();
    }
}
```

The declarations for this problem are:

```
Emotion a = new Happy();
Emotion b = new Sad();
Happy c = new Happy();
Happy d = new Sad();
```

```
Happy d = new Sad();
Sad e = new Sad();
```

If the code results in a compiler error, type Compiler Error. If the code results in a runtime error, type Runtime Error. If the code results in no output at all, type No Output. Otherwise, type out the output exactly as it would appear on the console when the code is run.

For this part, there are 7 separate problems below. Please answer each in isolation, i.e., no lines should affect the others.

1) e.yell();

2) c.smile();

3) a.yell();

4) b.cry();

5) a.smile();

6) b.smile();

7) c.yell();

Q5.3 Polymorphic Tracing Part 2 (Complex)

8 Points

Given the declarations below, you will be asked to trace the various pieces of code. For each portion of code, you are asked to type out the output precisely as it would be printed on the console (you may NOT run any of this code in Eclipse, you MUST trace this by hand).

The classes are again:

```
3  public abstract class Emotion {
4      public abstract void cry();
5      public abstract void laugh(int a);
6
7      public void smile() {
8          System.out.println("Turn that frown...");
9      }
10
11     public void yell() {
12         System.out.println("YAAAH!");
13     }
14 }
```

```
3  public class Happy extends Emotion {
```

```
    public void cry() {
        System.out.println("Finally... Matrix 4!");
    }

    public void laugh(int a) {
        System.out.println("They made Fast and Furious " + a);
        this.smile();
    }

    public void smile() {
        System.out.println("Ted Lasso is my hero");
    }
}
```

```
public class Sad extends Happy {

    public void cry() {
        super.cry();
        System.out.println("WHY end like that Game of Thrones?!?");
    }

    public void smile() {
        System.out.println("I had such unmet hope for the new Bill and Ted");
    }

    public void yell() {
        System.out.println("Not the Red Wedding!");
    }

    public void pout() {
        super.cry();
        System.out.println("Another year to wait for season 3...");
    }

    public void complain() {
        System.out.println("WHY cancel Firefly?");
    }
}
```

```
import java.util.ArrayList;

public class Feelings {
    private Emotion emo;
    private ArrayList<Happy> haps;

    public Feelings(Sad s) {
        this.emo = s;
        this.haps = new ArrayList<Happy>();
    }
}
```

The declarations for this problem are:

```
Emotion a = new Happy();
Emotion b = new Sad();
Happy c = new Happy();
Happy d = new Sad();
Sad e = new Sad();
```

If the code results in a compiler error, type Compiler Error. If the code results in a runtime error, type Runtime Error. If the code results in no output at all, type No Output. Otherwise, type out the output exactly as it would appear on the console when the code is run.

For this part, there are 8 separate problems below. Please answer each in isolation, i.e., no lines should affect the others.

1) Emotion varE = new Emotion();

2) b.laugh(9);

3) Sad varS = new Happy();

4) ((Emotion)a).smile();

5) ((Vanilla)a).smile();

6) ((Sad)a).pout();

7) d.cry();

8) b.complain();