CS 5004 Module 3 Worksheet
More Complex Data

Each worksheet is to help you remember and understand the materials contained in each module so that you can get the most out of your learning experience. We start by asking questions that are covered in the module but will also ask you to apply this knowledge. By taking the time to answer each question carefully, these worksheets can serve as the basis for review for the more formal assessments in this class.

1. In this module, we learned that a `switch` statement is an alternative to writing a long if-else if-else statement. Knowing this, rewrite the following as a `switch` statement. You may assume that the `number` variable has been declared as an `int` and has been assigned a value:

```
if (number == 0) {
  System.out.println("number is zero"):
} else if (number == 1 || number == 2) {
  System.out.println("number is tiny");
} else if (number == 3 || number == 4) {
  System.out.println("number is small");
} else if (number == 5) {
  System.out.println("number is bigger");
} else {
  System.out.println("number is greater than 5");
}
```

```
Switch(number) {
       Case 0:
              System.out.println("number is zero");
              break;
       case 1:
       case 2:
              System.out.println("number is tiny");
              break;
       case 3:
       case 4:
              System.out.println("number is small");
              break;
       case 5:
              System.out.println("number is bigger");
              break;
       default:
              System.out.println("number is greater than 5");
              break;
}
```

2. Consider the following enumeration that represents the days of a standard work week:

```
public enum WeekDay {
   MON(“Monday”),
   TUE(“Tuesday”),
   WED(“Wednsday”),
   THU(“Thursday”),
   FRI(“Friday”);

   Private String text;

   Private WeekDay(String text) {
        This.text = text;
   }

   @Override
   Public String toString() {
        Return text;
   }

}

WeekDay day = //some input value;
```

   a. We want to refactor the enumeration so that the switch statement shown on the right could be replaced with the simple print statement that is shown on the left. Use the information that is provided in the Java Tutorial on Enum Types that was linked from the module to rewrite the enumeration to make this possible.

```
switch (day) {
  case MON:
    System.out.println("Monday");
    break;
  case TUE:
    System.out.println("Tuesday");
    break;
  case WED:
    System.out.println("Wednesday");
    break;
  case THU:
    System.out.println("Thursday");
    break;
  case FRI:
    System.out.println("Friday");
    Break;
  default:
    throw new IllegalStateException();
```

```
System.out.println(day);
```

```
         }


public enum WeekDay {
   MON, TUE, WED, THU, FRI;
}

public class weekday {
        WeekDay  day;

        Public void printWeekDay() {
                switch (day) {
                        case MON:
                        System.out.println("Monday”);
                        break;

                         case TUE:
                        System.out.println("Tuesday”);
                        break;

                        case WED;
                         System.out.println("Wednesday");
                         break;

                        case THU;
                         System.out.println("Thursday");
                         break;

                        case FRI;
                         System.out.println("Friday");
                         break;

                    }
        }
}
```

b. Compare the switch statement to the simple print statement in the previous question. Which do you think is a better implementation for the `WeekDay` enumeration and why?

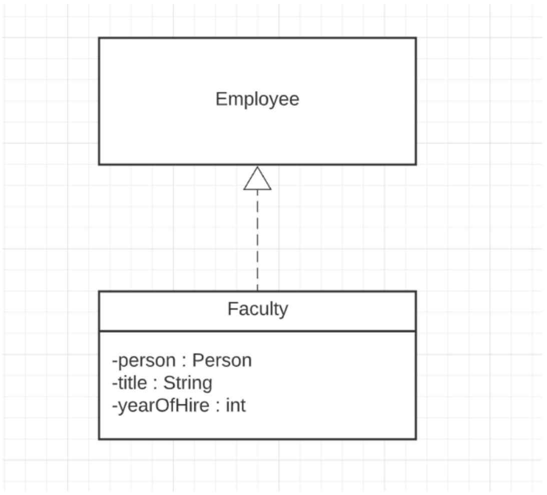
I think the switch and case implementation is better for weekday enumeration. It is because the switch and case implementation will have a default case, which specifies what happened if the input is not the case situations and throw a notification.

3. Explain the difference between *composition* and *inheritance*.

The composition is a design technique in which your class can have an instance of another class as a field of your class.
Inheritance is a mechanism under which one object can acquire the properties and behavior of the parent object by extending a class.

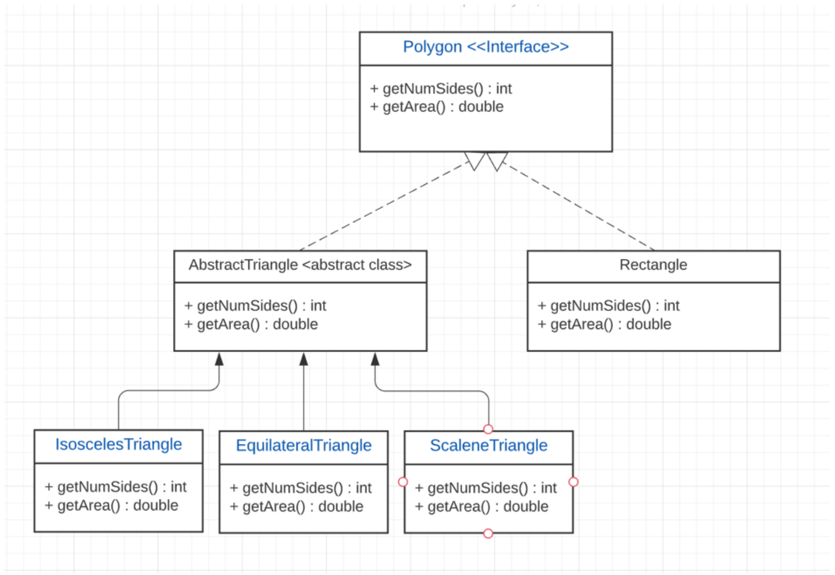
4. Given the following class definition, draw an appropriate UML diagram. Do not worry any details that have been omitted:

```
public class Faculty implements Employee {
  private Person person
  private String title;
  private int yearOfHire;

  // lost of code omitted for simplicity
}
```



5. Suppose that you wanted to create the following classes: `Polygon`, `Triangle`, `IsoscelesTriangle`, `EquilateralTriangle`, `ScaleneTriangle`, `Rectangle` where we wanted to have methods like `getNumSides()` and `getArea()`.

   a. Draw an appropriate class hierarchy that would capture the appropriate inheritance relationships. Distinguish between interfaces, abstract classes, and concrete classes (as appropriate).



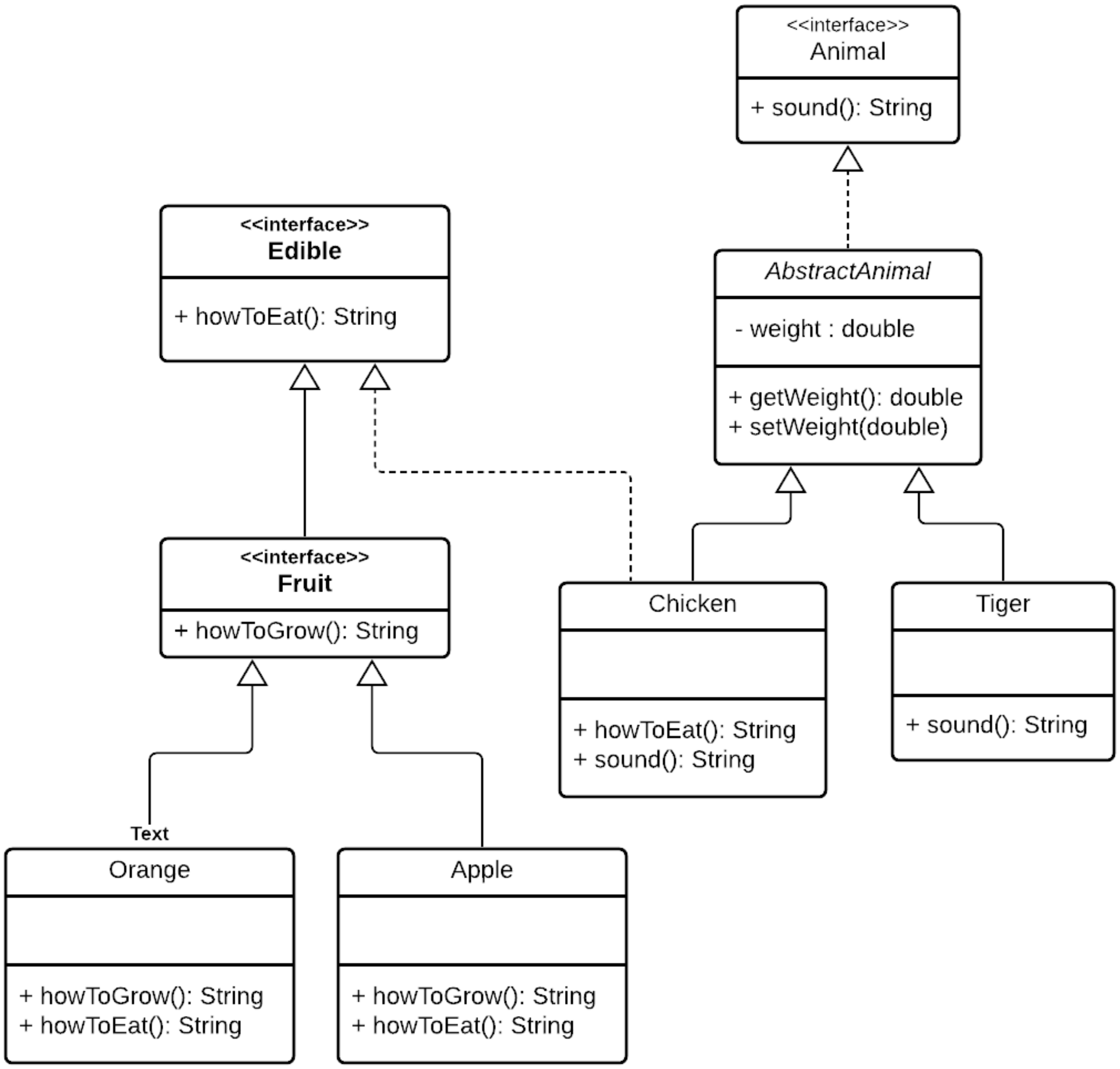
   b. Based on the class hierarchy you drew in part (a), declare the `Rectangle` class. You do not need to include any of the fields or methods.

```
Public class Rectangle implement Polygon {
        Public int getNumSides() {
        }

        Public double getArea() {
        }
    }
```

   c. Based on the class hierarchy you drew in part (a), declare the `ScaleneTriangle` class. You do not need to include any of the fields or methods.

```
Public class ScaleneTriangle extends AbstractTriangle {
        Super();
        Public int getNumSides() {
        }

        Public double getArea() {
        }
```

6. Consider the following class diagram:



For each of the following, indicate what type(s) of variable could be used to store any of the listed types, or indicate that it is not possible:

Apple or Orange	Edible, Fruit, Apple, Orange
Apple or Orange, but not Chicken	Fruit, Apple, Orange
Apple or Orange or Chicken	Edible
Chicken or Tiger	Edible, Animal, AbstractAnimal
Tiger or Apple	Not Possible
Chicken or Orange	Edible
Apple or Orange or Chicken or Tiger	Not possible