Setup: Here are the names of an assortment of classes and an interface: TextBook, Person, CS115Class, Student, Readable (Interface), Professor,

The relationships between these classes is as follows:
A CS110Class has a Professor. (Aggregation)
A CS110Class has a Student (Aggregation)
A Professor is a Person (Inheritance)
A Student is a Person (Inheritance)
A Student uses a TextBook (Dependency)
A TextBook is Readable (adjective) (Interface)

The following UML diagram shows the relationship between these classes.
1. Attributes and Methods for Classes
For the classes and the interface defined in problem 1, show where, how, and why you would define the following attributes and/or methods in one of the classes or the interface. Include the following for each of them: the visibility modifier, the use of static or not, the use of final or not, the most appropriate data type, and a suitable name. Update the UML Diagram Above with each attribute/method.

a. grade point average
In class/interface: ________________________________
Line of Java code to use:

Explain:

b. number of pages
In class/interface: ________________________________
Line of Java code to use:

Explain:

c. read method that takes two integer values for starting page and ending page and returns nothing
In class/interface: ________________________________
Line of Java code to use:

Explain:

d. a public constant defining “hard to read” as true or false
In class/interface: ________________________________
Line of Java code to use:

Explain:
2. Classes, Interfaces, Inheritance, and Polymorphism

Using the UML diagram and attributes/methods from question 1 and 2, answer if each of these lines of Java code in class UMLMain is valid or not. Explain your answer!!
(Note: Assume that UMLMain is not in the same package or directory as the other classes.)

```java
public class UMLMain
{
    public static void main(String[] args)
    {
        Person p;
        Student s;
        Readable r;
        Textbook t;

        p = new Professor(); // a. Valid? ____________
        s = new Professor(); // b. Valid? ____________
        p = new Student(); // c. Valid? ____________
        s = new Student(); // d. Valid? ____________
        s = new Person(); // e. Valid? ____________
        r = new TextBook(); // f. Valid? ____________
        t = new Readable(); // g. Valid? ____________
        r = new Readable(); // h. Valid? ____________
        t.read(1, 25); // i. Valid? ____________
        double gpa = s.gradePointAverage; // j. Valid? ____________
    }
}
```
3. Exceptions and Exception Handling

Study the following Java code and answer the questions below

```java
public class TestException // 01
{
    public static void main(String[] args) // 02
    {
        String [] band = new String [4]; // 05
        try { // 06
            band[0] = "John Lennon"; // 08
            band[2] = "Paul McCartney"; // 09
            band[4] = null; // 10
            band[1] = "Ringo Starr"; // 11
            band[3] = "George Harrison"; // 12
        } // 13
        catch (ArrayIndexOutOfBoundsException e) { // 14
            System.out.println("Only four Beatles"); // 15
        } // 16
        catch (NullPointerException e) { // 17
            System.out.println("Bad Coding Practice"); // 18
        } // 19
        finally // 20
        { // 21
            for (String beatle : band) // 22
                System.out.println(beatle); // 24
        } // 25
    } // 26
} // 27
```

(a) Do we need to add a “throws exception” clause to the definition of main? YES/NO _______
(b) If so, why? If not, why not?

(c) When run, will any line above throw an exception? YES/NO _______
(d) If c is YES, which line number, what exception will it throw, and why?
   Line Number: ______ Exception: _____________________________________________
   Why? ______________________________________________________________________

(e) If c is YES, which line number will execute immediately after the exception occurs? _____
(f) When run, what will be printed by the main method?
4. Interfaces: Know the client code from the practice midterm - Suppose class PhoneNumber has the following API:

```java
public class PhoneNumber
    PhoneNumber(int areaCode, int exchange, int localNumber)
int getAreaCode()     // area code of number (3 digits)
void setAreaCode(int newAreaCode) // replace old area code with new one
int getExchange()     // exchange of number (3 digits)
int localNumber()     // local number (4 digits)
boolean isValid()     // true if all 3 numbers fit in the #digits allowed
String toString()     // a String such as "617-354-5500"
```

Continuing with PhoneNumber.java. Turn the API methods for PhoneNumber (other than toString(), an Object method) into an interface Phone.java. Show the full source code for Phone.java. How do you make PhoneNumber.java relate to the Phone interface?
5. In Project p4, we read all the words (Strings with no internal spaces) from words.txt into an ArrayList<String> words. Write a helper method named hasDuplicates that accepts words as a parameter and checks whether or not there are duplicates among these given words, using the same duplicate detection method used for the “codes” in the project. The method hasDuplicates should return true if any duplicates exist, else false.

```java
public class SeaCreature {
    public void method1() {
        System.out.println("creature 1");
    }
    public void method2() {
        System.out.println("creature 2");
    }
    public String toString() {
        return "ocean-dwelling";
    }
}

public class Mammal extends SeaCreature {
    public void method1() {
        System.out.println("warm-blooded");
    }
}
// More below...
```
public class Whale extends Mammal {
    public void method1() {
        System.out.println("spout");
    }
    public String toString() {
        return "BIG!";
    }
}

class SeaCreature {
    public void method2() {
        System.out.println("tentacles");
    }
    public String toString() {
        return "squid";
    }
}

// Questions below
6. (con’t) Here is some client code for those classes:

```java
ArrayList<SeaCreature> animals = new ArrayList<SeaCreature>();
animals.add(new Whale());
animals.add(new Squid());
animals.add(new Mammal());
animals.add(new SeaCreature());
```

a. Explain why we can add a Whale to an ArrayList<SeaCreature>, given that add expects an argument of type SeaCreature.

b. What is printed out by

```java
for (int i=0; i< animals.size(); i++) {
    animals.get(i).method1();
}
```

c. What is printed out by

```java
System.out.println(animals);
```