Name ____ Section Number ________

CS110 Exam #3 *** PLEASE TURN OFF ALL CELL PHONES*** Practice
Sections All Bob Wilson

You may only use your crib sheet (one HANDWRITTEN 8-1/2 x 11 page both sides). You will have all 90 minutes until the start of the next class period. Spend only about one minute per point on each question to complete the exam on time.

1. (30 Points) Here are the names of an assortment of classes and an interface:
TextBook, Person, CS110Class, Student, Readable (Interface), Professor,

a. Write the "standard statements", e.g. has-a, for each related class or interface pair and indicate the name of the relationship.

b. Draw a UML class diagram that shows the inheritance, interface, aggregation, and/or dependence relationships between these classes and the one interface. Leave enough room to fill in the attributes and/or methods from problem 2.

c. As you do problem 2, add the four attributes/methods to your above UML diagram.

2. Attributes and Methods for Classes (20 Points)
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.

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:

DON"T FORGET TO ADD THE ABOVE TO YOUR UML DIAGRAM IN PROBLEM 1

3. Classes, Interfaces, Inheritance, and Polymorphism (30 Points)

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.)

```
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? ____________

  }
}
```

4. Exceptions and Exception Handling (20 Points)

Study the following Java code and answer the questions below

```
public class TestException                             // 01
{                                                      // 02
   public static void main(String[] args)              // 03
   {                                                   // 04
     String [] band = new String [4];                  // 05
                                                       // 06
     try {                                             // 07
          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?

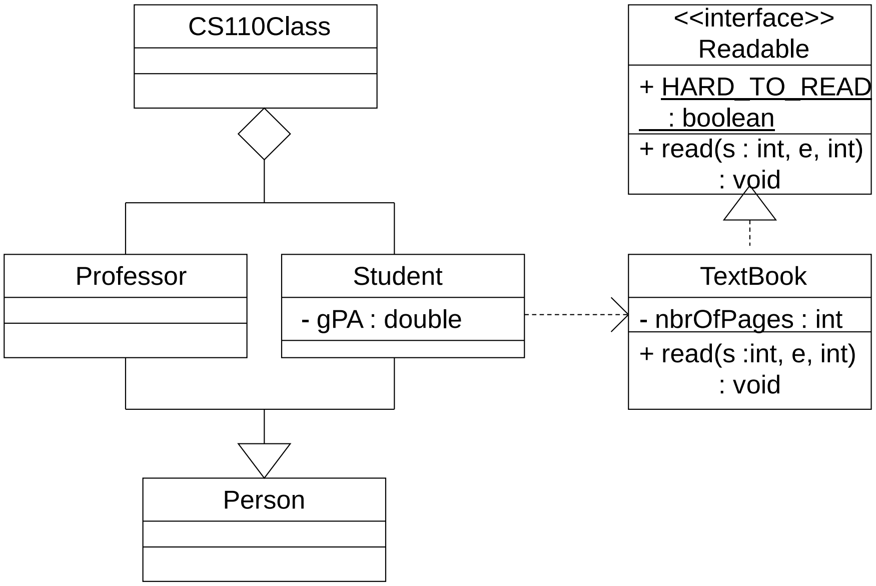
Answer Key:

1.

a. Write the “standard statements”, e.g. has-a, for each related class or interface pair and indicate the name of the relationship.

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

b. UML Diagram:



Note: The above seem to be the most logical of the relationships between the classes and the interface. However, if your rationale for identification of relationships is different, the UML class diagram could be different.

2. Attributes and Methods for Classes

a. grade point average
In class/interface: Student
Line of Java code to use:
private double gradePointAverage;

Explain: Students have grade point averages so it is a Student attribute. It is an instance variable. Each student has a unique grade point average so static and final are not used. It is best represented by a double value. A setter and getter method would be provided.

b. number of pages
In class/interface: TextBook
Line of Java code to use:
private int numberOfPages;

Explain: Textbooks have pages so it is a TextBook attribute. It is an instance variable. Each textbook has a unique number of pages so static and final are not used. It is best represented by an int value. A setter and getter method would be provided.

c. read method
In class/interface: Readable
Line of Java code to use:
public void read (int startingPage, int EndingPage); (Note: No { } in an interface.)

Explain: Each class implementing Readable probably would have a read method. It is an instance method. Each Readable object is uniquely read so static and final are not used. Nothing is returned, the return type is shown as void. The textbook class would need to have code for this method to implement the Readable interface.

d. a public constant defining "hard to read" as true or false
In class/interface: Readable
Line of Java code to use:
public static final boolean HARD_TO_READ = true;

Explain: Readable classes could be either hard to read or easy to read. It is an interface level constant best represented by a boolean value. This constant could be used in Classes that implement Readable or classes that deal with classes that implement Readable.

3. Classes, Interfaces, Inheritance, and Polymorphism

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

    p = new Professor();                // a. Valid? _Yes________
                              // Widening Conversion
    s = new Professor();                // b. Valid? _No_________
                              // Incompatible Classes (Siblings)
    p = new Student();                  // c. Valid? _Yes________
                              // Widening Conversion
    s = new Student();                  // d. Valid? _Yes________
                              // Normal instantiation
    s = new Person();                   // e. Valid? _No________
                              // Incompatible Classes (Narrowing)
    r = new TextBook();                 // f. Valid? _Yes________
                              // Widening Conversion
    t = new Readable();                 // g. Valid? _No_________
                              // Can't instantiate an interface
    r = new Readable();                 // h. Valid? _No_________
                              // Can't instantiate an interface
    t.read(1, 25);                      // i. Valid? _Yes________
                              // Normal method call
    double gpa = s.gradePointAverage;   // j. Valid? _No_________
  }                           // An attribute should be private
}
```

4. Exceptions and Exception Handling

(a) NO

(b) We catch the exception in a "try" statement with "catch" clauses

(c) YES

(d) Line number 10 will throw an ArrayOutOfBoundsException for index 4 > 3

(e) Line number 14

(f) The main method prints:

```
> java TestException
Only four Beatles   (in the catch cause for ArrayOutOfBounds)
John Lennon
null                (code never got to initialize this element)
Paul McCartney
null                (code never got to initialize this element)
```