The City College of New York
Department of Compute Science – CSc 221: Software Design Laboratory – 2nd Exam

Name:_____ ID:_____

**Q1) (6p) List three differences between *inheritance* and *composition* in Java?**

1. _____When the relation is an is-a relation and not has-a relation__(1-to-1)_____
2. _____When the relation is a tight relation between objects__(1 to many)_____
3. _____When we'd like to extend the functionalities of an already existing class_____
4. _____Can access protected members_____
5. _____ When the design of the classes calls for a hierarchical design_____

_____ When a common object is needed to refer to multiple other types (polymorphism)_____

**Q2) (6p) List three similarities or differences between a Java *abstract* class and a Java *Interface*?**

1. _____Both are Java's polymorphism implementation _____
2. _____Can only extend one abstract class, Can implement multiple interfaces _____
3. _____Interface are declared via the interface keyword_____
4. _____Abstract classes can have methods that are abstract or with implementation_____
5. _____Interfaces are pure abstract classes, Abstract classes are declared via abstract keyword _____
6. _____Neither one can be instantiated, Abstract classes are extended but interfaces are implemented_____

**Q3) (10p) When designing a new Java class, it is recommended that it overrides some of the *Object* class' methods. One such method is *equals*. Briefly describe this method and discuss when and why it should be overridden.**

- This method compares two objects for equality
- The default implementation uses the default hashcode() implementation which is not for accurate object comparison, this method aught to be precisely defined
- Since the parameter is of type object, it is not precise. The method should check the instance of the object parameter as well

**Q4) (10p) JavaFX applications consist of several components including the *FXML* file and the *Controller* file. Briefly explain each of these files including the programming language(s) used by each one.**

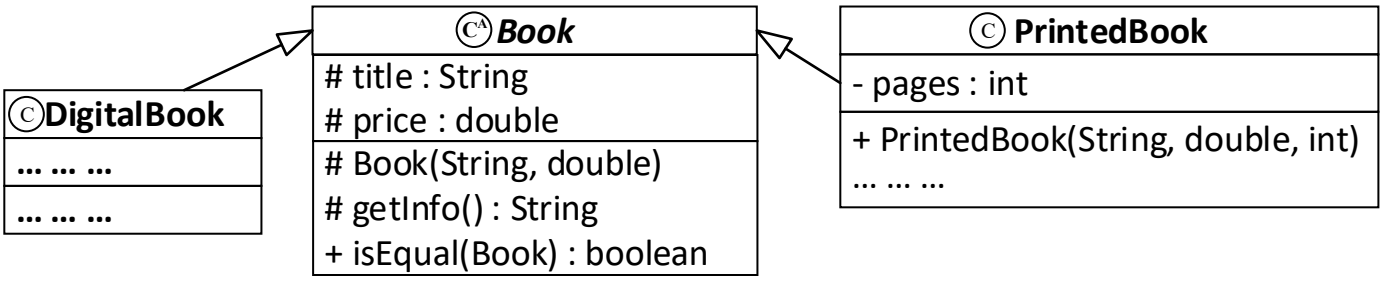
✓ **FXML file**

- FXML (FX Markup Language)
- An XML vocabulary for defining and arranging JavaFX GUI controls without writing any Java code (layout, colors, dimensions …)
- Separates the GUI from the interface

✓ **Controller file**

- A java class which extends
- GUI's event handlers are defined in a so-called controller class
-

Name:_____ ID:_____




**Q5) (21p)Implement the Abstract class *Book*. The method *getInfo*() returns the values of *title* and *price* separated by a colon (:). Two *Books* are equal if their prices are equal. The symbol # indicates a *protected* member.**

```
public abstract class Book { // 3p
    protected String title; // 2p
    protected  double price; // 2p
    protected Book(String title, double price) {  // 7p
        this.title = title;
        this.price = price;
    }
    protected String getInfo() { // 3p
        return title + ":" + price;
    }
    public boolean isEqual(Book other) { // 4p
        return price == other.price;
    }
}
```

**A. (25p) Write the implementation of the class PrintedBook. Override the *isEqual*() method such that it checks the value of the *pages* variable as well as *title* and *price*. Note that you must reuse the base class' *isEqual*().**

```
public class PrintedBook extends Book { // 3p
    private int pages; // 1p
    public PrintedBook(String title, double price, int pages) {  // 2p
        super(title, price); // 2p
        this.pages = pages; // 1p
    }
    public boolean isEqual(Book other) { // 2p

        if( other instanceof PrintedBook)        // 3p
            return                              // 2p
                  super.isEqual(other) && // 2p
                  title==other.title &&   // 3p
                  pages==((PrintedBook)other).pages; // 3p
        return false; // 1p    } }
```

Name:_____ ID:_____

**B. (22p) Complete the following function such that it returns an *ArrayList* containing instances of type *PrintedBook* that match the parameter *pages*. Note that this function is part of a separate class called *TestClass* and NOT part of either of the classes shown in the diagram above.**

```
public class TestClass {

    public static void main(String args[]){
    }
    public ArrayList<PrintedBook> filter(ArrayList<Book> bookList, int pages) {

        ArrayList<PrintedBook> retList;       // 2p

        retList = new ArrayList();            // 2p

        for(Book b : bookList) {              // 3p
            if(b instanceof PrintedBook) {    // 3p
             PrintedBook book = (PrintedBook)b; // 3p

            if(book.pgetPages() == pages)     // 3p
                retList.add(book);      // 3p
        }
    }
        return retList;                       // 3p
}
```