Assignment 2

Exercise 1

1. The implementation of inheritance always involves writing a few less lines.
A. False

2. The following class declaration is incorrect:
Public final class Class extends OtherClass {...}
A. False

3. Inheritance is only useful if specialization is used. In fact, specializing we inherit in the subclass (or subclasses) members of the superclass that we must not rewrite. Instead with the generalization we create an extra class, and then we write more code.
A. False

4. The super keyword allows you to call superclass methods and constructors. The keyword this allows you to call methods and constructors of the same class.
A. True

5. Mutliple inheritance does not exist in Java because it does not exist in reality.
A. False (It is implemented using Interfaces)

6. A subclass is bigger than a superclass (in the sense that it usually adds new features and functionality compared to the superclass).
A. True

7. Suppose we develop an application to manage a soccer tournament. There is inheritance derived from specialization between the Team and Player Classes.
A. True

8. Suppose we develop an application to manage a soccer tournament. There can inheritance derived from generalization between the Team and Player Classes.
A. True

9.In general, if we had two classes, Father and Son, there would be no inheritance between these two classes.
A. False

10. An interface can extend an abstract class and an interface.
A. False (Interface can extend other interface but not a class)

Exercise 2:
/src/exercise2/Book.java
/src/exercise2/BookTest.java

Exercise 3:
/src/exercise3/Book.java
/src/exercise3/Bookcase.java
/src/exercise3/Bookstore.java
/src/exercise3/BookstoreTest.java

Exercise 4:
/src/exercise4/Album.java
/src/exercise4/Book.java
/src/exercise4/Solution.java
/src/exercise4/SolutionAlbum.java
/src/exercise4/SolutionBook.java
/src/exercise4/SolutionTest.java

Exercise 5:
/src/exercise5/Assignment.java
/src/exercise5/Course.java
/src/exercise5/CourseRecord.java
/src/exercise5/Student.java

Exercise 6:
UML Diagram

Store

-address: String
-fax: double

+addFurniture(f: Furniture): void
+removeFurniture(f: Furniture): void
+placeOrder(): void

buys

1

*

Order

-orderId: int
-date: Date
- quantities []

+printDetails(): void

Factory

+addFurniture(f: Furniture): void
+getFurniture(): ArrayList

1..*

furnitures*

Furniture

-name: String

+getName(): String
+setName(String n): void
+addPiece(piece: Piece): void
+removePiece(piece: Piece): void

1..*

Piece

-type: String
-reference: String
-price: double

+addComponent(c: Component): void
+removeComponent(c: Component): void

1..*

1..*

Component

-id: double
-price: double
-type: String