Problems corresponding to Unit 1: Revision of the basics of OOP.

1. Declaring, creating and using objects from user-written classes: Copy the entity classes Player, BankAccount and Person into your own folder, compile and inspect each, then write an test application which creates a player object, a bank account and a Person object. Display the details of all the objects in your test class. Build on this by making the Person class an abstract class, the Player class an extension of the Person class and the Player class an aggregate class that contains a BankAccount attribute. Modify you classes to make them fit as a cohesive system. Test the system using an appropriate driver class.
2. Adding extra attributes to an instantiable class, boolean attributes and using a List: Add a boolean attribute ‘online’ to your Player class, with a boolean isOnline() method, a setOnline(boolean) method and any necessary changes to the constructors and toString(). Write an application which declares a LinkedList myPlayers of Players, asks you to enter names for two players (an online player and one who is not online), sets up all the details for the players and adds them to myPlayers, then displays the details of both in the simplest way possible using a message dialog and the toString() method. See to it that your toString() method displays ‘online’ or ‘not online’ rather than ‘true’ or ‘false’.
3. Retrieving information from files, and the algorithm for finding the highest: Add a section to your solution to the previous exercise which will write the list of Players out to a file. Then write a separate application which will read back the players and find and display the name of the player who has the highest score. Write a pseudocode algorithm for this first.
4. Writing instantiable classes and using them Write a GoodGuy and a BadGuy entity class, each of which has the attributes name, strength, ammo and isAlive (boolean), with the usual methods. Write an application which uses them as follows: your application should declare and create a GoodGuy and a BadGuy. It should contain two buttons “Good shoots Bad” and “Bad Shoots Good”, and a text area which shows the state of your two guys. Consider what should happen if Good shoots Bad: you will need to add methods to both entity classes to implement appropriate actions (strength reduced, ammo reduced or whatever). Remember that a good method should only do one thing: if a method is trying to do more than one thing, you need to write two methods.

[Optional Exercises]

1. Declaring, creating and using objects from the Java API: Review the API classes JTextArea, Color and Font. Write a class which declares and creates each of the following:

a JTextArea,

two Colors of your choice, one for foreground and one for background

a monospaced 16-point plain Font:

Set the foreground, background and font for the text area, append some text to it and display it in a message dialog. Advanced extra: display a JColorChooser object first, and allow the user to pick a colour for the background, which should then be set up automatically. You will need to see what methods are available for a JColorChooser in order to do this successfully.

1. Identifying classes and methods: Draw up a list of the classes you think might be needed for a computer system to model either a Chess game or a Poker game: for two of these classes, list the class-specific methods that it would need (other than accessors, mutators, constructors and toString()).