SCC.311 Coursework Part A Specification
Due: Friday 4pm in Week 3 via Moodle

The goal of this coursework stage is to build a simple client-server system that invokes a method using Java RMI. This system represents the start of an online auction system, which you will develop into more advanced solutions over the remainder of this module. An auction system is one in which a user can advertise items for sale and other users can make bids on that item; when the auction closes, the highest bidding user will get the item.

This coursework stage is marked using an automated test system, so the Java interface that you use must **exactly** match the specifications given here. Please read the submission guidelines at the end of this specification for more detail.

This coursework has two levels that you can attempt. You should submit only the highest level that you have completed; if you finish level 2, you do not need to also submit level 1.Note that developing only Level 1 will gain significantly fewer marks (i.e., 6/17) for **this stage** of the coursework, but **does not prevent you** from gaining full marks in the subsequent coursework stages.

Level 1: Invocation (6 marks)
Build an RMI server that offers the below exact interface, where the interface **is not in a package**:

```
public interface Auction extends Remote {
public AuctionItem getSpec(int itemID) throws RemoteException;
}
```

The AuctionItem class should be declared in a separate class file as:

```
public class AuctionItem implements java.io.Serializable {
int itemID;
String name;
String description;
int highestBid;
}
```

The getSpec() call should return the details of an auctioned item that has the identifier itemID. You must also build a client that invokes the above method on the server using RMI, and displays the return values to the user. A basic command-line client is sufficient, but must allow the user to enter the itemId details (this should not be hard-coded). For your server, at this stage, it is sufficient to use a hard-coded set of auction items.

Level 2: Encryption (11 points)
This level should have all the functionality of Level 1, but must provide the following interface instead:

```
public interface Auction extends Remote {
public SealedObject getSpec(int itemID) throws RemoteException;
}
```

This version of the interface supports encryption, and you should not offer the level 1 interface as part of this solution. For encryption support, you should use the Java SealedObject class, a mechanism that allows you to encrypt and decrypt Java objects. When doing this you should use AES encryption and AES keys rather than plain passwords.

You do not need to develop a key distribution mechanism for this stage of the coursework; you can simply store your session key on disk in a location that both the client and server can access. You will need to generate a shared key (see the KeyGenerator class) which you can store in a file.

Coursework submission instructions are below.

Your coursework will be marked using an automated test system. For the test system to work properly, your submission must be contained in a zip file and have the following:

1. A shell script called server.sh, **in the root directory of your submission**, which performs any set of operations necessary to get your server running.
2. An RMI service advertised with the name "Auction"
3. An RMI interface which **exactly matches** the specification given in this document.
4. Your entire system must be up and running within 5 seconds (our automated test is launched 5 seconds after running your server.sh script).
5. If you attempt level 2, a directory called 'keys' which contains an AES key stored in a file called testKey.aes

Not following the above instructions will lead to a loss of marks.

Mark Scheme

Level 1

Client invocation and display — 3 marks
Server interface — 3 marks

Level 2

Achieving this level implies that you will gain all of the marks for Level 1
Key creation and use — 3 marks
A working encryption/decryption (automated tester is able to decrypt the response which matches the expected spec) — 8 marks