1. Short answers (15 points). Give very short (1 to 2 sentences for each issue) answers to the following questions. **Longer responses to these questions will not be read.**

   (a) What does the value of the rmi.server.codebase property mean when used in an RMI program. Be specific.

   Answer:

   The codebase is the location from which the class files for objects created within a given JVM can be downloaded.

   (b) In class we compared the performance of server architectures with single or multiple threads, using blocking or nonblocking I/O. For the examples shown, which approach had the best performance characteristics?

   Answer:

   multithreaded, non-block I/O.

   (c) What general strategy does the Erlang programming language use to avoid synchronization issues?

   Answer:

   Nearly all data is immutable.

2. RMI. (20 points). The following code is intended to implement a rudimentary game called Ping-Pong. In a PingPong game there are two Players and a Referee. The referee gives a Ball to one of the players by calling Player.serve(). That player tosses the Ball to the other Player, by calling the other Player's Player.catchIt() method. The receiving player calls Ball.beforeCatch(), which mimics the time needed for the Ball to arrive. Next, the receiving Player tries to catch the Ball. If the Player succeeds, it tosses the Ball back by calling its own Player.tossIt() method. If the Player fails to catch the Ball, it notifies the Referee by calling Referee.dropped(). The Referee.dropped() method updates the game score and then gives the Ball to one of the Players for the next serve.

   Assume that the Players and the Referee code each run on different machines. The Player instances must register with an RMI registry to indicate that they are available to play. Assume that the registry has been started before the Players begin executing. When the Referee instance runs it must retrieve two Players from the registry and then coordinate play between them.

   a) Fill in the main() methods below to complete the implementation and b) Assuming there are 2 Players and 1 Referee running on different machines and that each machine has only the minimum files needed to compile the code running on that machine, what files must by made available for download via a webserver?

```
/* Referee.java */
package cmsc433S11.FinalExam.PingPong.common;
public interface Referee extends Remote {
   void dropped (Player loser) throws RemoteException;
}
```

```
/* Player.java */
package cmsc433S11.FinalExam.PingPong.common;
public interface Player extends Remote {
 void init (Referee ref, Player opp) throws RemoteException;
 void serve(Ball ball) throws RemoteException;
 void tossIt(Ball ball) throws RemoteException;
 void catchIt(Ball ball) throws RemoteException;
}
```

```
/*Ball.java */
package cmsc433S11.FinalExam.PingPong.common;
public interface Ball extends Serializable {
  void beforeCatch();
}
```

```
/* RefereeImpl.java */
package cmsc433S11.FinalExam.PingPong.Referee;
public class RefereeImpl implements Referee {
  Player p1, p2;
  int p1Score, p2Score, max = 15;
  boolean p1Serves = true;

 public void dropped(Player loser) throws RemoteException {
  p1Serves = (loser.equals(p1));
  if (p1Serves) {p1Score++;} else {p2Score++;}
 }

 void go(Player p1, Player p2) {
  try {
   this.p1 = p1; this.p2 = p2;
   Ball ball = new VarSpeedBall();
   p1.init(this, p2);
   p2.init(this, p1);
   while (p1Score < max && p2Score < max) {
    if (p1Serves) {p1.serve(ball);} else {p2.serve(ball);}
   }
  } catch (Exception e) {}
 }

 public static void main(String[] args) {
```

Answer:

```
      if (System.getSecurityManager() == null) {
       System.setSecurityManager(new RMISecurityManager());
      }
      try {
       RefereeImpl ref = new RefereeImpl();
       UnicastRemoteObject.exportObject(ref, 0);
       Registry registry = LocateRegistry.getRegistry(/*host*/, /*portNumber*/);
       Player p1 = (Player) registry.lookup(/*player1*/);
       Player p2 = (Player) registry.lookup(/*player2*/);
       ref.go(p1, p2);
      } catch (Exception e) {}
     }









}
```

```
/*PlayerImpl.java*/
package cmsc433S11.FinalExam.PingPong.Player;
public class PlayerImpl implements Player {
 Player stub, opp;
 Referee ref;

 public void setProxy (Player myStub) {
  this.stub = myStub;
 }

 public void init (Referee ref, Player opp) throws RemoteException {
  this.ref = ref; this.opp = opp;
 }

 public void tossIt(Ball ball) throws RemoteException {
  opp.catchIt(ball);
 }

 public void catchIt(Ball ball) throws RemoteException {
   ball.beforeCatch();
   if (new Random().nextFloat() > .5) {tossIt(ball);}
   else { ref.dropped(stub); }
 }

 public void serve(Ball ball) throws RemoteException {
  tossIt(ball);
 }

 public static void main(String[] args) {
```

Answer:

```
      if (System.getSecurityManager() == null) {
       System.setSecurityManager(new RMISecurityManager());
      }
      try {
       PlayerImpl me = new PlayerImpl();
       Player stub = (Player) UnicastRemoteObject.exportObject(me, 0);
       me.setProxy(stub);
       Registry registry = LocateRegistry.getRegistry(/*host*/, /*portNumber*/);
       registry.rebind(/*playerID*/, stub);
      } catch (Exception e) {}
     }




}
```