1. Design a class Player. A Player has a name and stores an arbitrary long sequence of numbers inside. It can calculate its score (the sum of all the numbers inside) and can determine if it’s a winner. A Player can also display itself (name, scores and whether it’s a winner or not). Let’s that for the purpose of some arbitrary game a Player is a winner if its score (sum of all the numbers it has) is at least 21. Write your own class or finish the class below so it works as shown when compiled and run:

```bash
-javac Player.java
-java Player
Larry Bird: [] false
Larry Bird: [3] false
Larry Bird: [3, 12] false
Larry Bird: [3, 12, 9] true

You can define your own class or finish the one shown here:

```java
import java.util.*;

class Player extends Object {  
  String name;
  -----(String name) {  
    ----.---- = ----;  
  }
  public String --------() {  
    return this.name + ": " + ----.toString() + " " + --------();
  }
  int score() {  
    int --- = 0;
    for (-------- i : this)
      --- += i;
    return ---;
  }
  -------- isWinner() {  
    return --------() >= 21;
  }
  public static void main(String[] args) {
    Player a = new Player("Larry Bird");
    System.out.println(a);
    a.add(3);
    System.out.println(a);
    a.add(12);
    System.out.println(a);
    a.add(9);
    System.out.println(a);
  }
}
```
2. Design a class Dice. Objects of this type have a method shake that produces a random integer between 1 and 6 inclusive. Write your own class or finish the class below so it works as shown when compiled and run:

```
-bash-3.2$ javac Dice.java
-bash-3.2$ java Dice
3 5 6 1 4 3 2 5 4 4
-bash-3.2$ java Dice
3 2 1 6 5 4 6 4 1
-bash-3.2$ java Dice
4 3 2 1 3 6 5 6 5 6
-bash-3.2$ java Dice
1 6 3 3 1 4 5 1 4 2
-bash-3.2$ java Dice
6 3 5 1 4 1 5 6 6 5
```

You can define your own class or finish the one shown here:

```java
----- Dice {
  ----- () {
    ------ (int) (-----.random() * 6 + 1);
  }
  public static void main(String[] args) {
    Dice d = new Dice();
    for (int i = 0; i < 10; i++)
      System.out.print(d.shake() + " ");
  }
}
```

3. Design a class Game. Objects of this kind run a simple game between a number of Players (as defined above). You can finish the class below or develop your own. The functionality shown can be summarized as follows: the number of players in the game is indicated on the command line. The game knows about the players and has a dice inside. In the game the players take turns throwing the dice. The game ends with the first player that qualifies as a winner. As an example let’s consider the following session:

```
-bash-3.2$ java Game 3
Player 0: [6] false
Player 1: [6] false
Player 2: [3] false
Player 0: [6, 6] false
Player 1: [6, 4] false
Player 2: [3, 3] false
Player 0: [6, 6, 4] false
Player 1: [6, 4, 1] false
Player 2: [3, 3, 3] false
Player 0: [6, 6, 4, 6] true
```

The game is started with 3 players. Player 0 is the first one to reach or surpass 21. Since Player 0 is a winner the game stops. In this example the first player won. This doesn’t have to be the case necessarily. In the example below (same code is run again also with an argument of 3) the winner is Player 2 (the last player). Any player can win, since the dice is completely random.
bash-3.2$ java Game 3
Player 0: [2] false
Player 1: [3] false
Player 2: [2] false
Player 0: [2, 6] false
Player 1: [3, 6] false
Player 2: [2, 6] false
Player 0: [2, 6, 2] false
Player 1: [3, 6, 4] false
Player 2: [2, 6, 6] false
Player 0: [2, 6, 2, 1] false
Player 1: [3, 6, 4, 4] false
Player 2: [2, 6, 6, 6] false
Player 0: [2, 6, 2, 1, 2] false
Player 1: [3, 6, 4, 4, 1] false
Player 2: [2, 6, 6, 6, 1] true
bash-3.2$

Develop your own class or reconstruct mine:

```java
import java.util.*;

class Game {
    ArrayList<Player> players;
    Dice d = Dice();

    Game(ArrayList<Player> players) {
        this.players = players;
    }

    void start() {
        for (Player p : players) {
            System.out.println(p);
            if (p.is()) {
                System.out.println(-);
            }
        }
    }

    public static void main(String[] args) {
        ArrayList<Player> players = new ArrayList<Player>();
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            players.add(new Player("Player " + i));
        Game a = new Game(players);
        a.start();
    }
}
```
4. Now without making any changes to Game.java, Dice.java and Player.java (in other words you can’t modify the source code for those three classes) write code that would give you a game that would run as follows (I added two classes, one of which is called Jeu; Jeu means Game in French. The game works as it was working before, we only change the criterion for being a winner: instead of scoring 21 or more the winner is the first player that has a score divisible by 7). Here’s a game in which Player 0 wins:

```
bash-3.2$ javac Jeu.java
bash-3.2$ java Jeu 3
Player 0: [6] false
Player 1: [3] false
Player 2: [5] false
Player 0: [6, 3] false
Player 1: [3, 1] false
Player 2: [5, 4] false
Player 0: [6, 3, 3] false
Player 1: [3, 1, 6] false
Player 2: [5, 4, 2] false
Player 0: [6, 3, 3, 2] true
bash-3.2$ java Jeu 3
Player 0: [3] false
Player 1: [2] false
Player 2: [3] false
Player 0: [3, 3] false
Player 1: [2, 3] false
Player 2: [3, 2] false
Player 0: [3, 3, 6] false
Player 1: [2, 3, 1] false
Player 2: [3, 2, 2] true
bash-3.2$
```

Note that in the first game all players score more than 7 and Player 0 wins. Second time Player 2 hits 7 before the others and wins. Any player can win, the game is random. Here’s all of my Jeu.java from which you can infer what’s the name of the other class I defined1.

```java
import java.util.*;

class Jeu {
    public static void main(String[] args) {
        ArrayList<Player> players = new ArrayList<Player>();
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            players.add(new Joueur("Player " + i));
        Game a = new Game(players);
        a.start();
    }
}
```

1 Joueur means Player in French.
You have to admit Jeu.java is unbelievably similar to Game.java (differs by only one word). Provide Joueur.java or do whatever else you want (short of touching the source code of the three classes Player, Game and Dice) to produce the same effect. The space below is sufficient to define Joueur.java:

5. Starting from Player.java, Dice.java and Game.java and without making any change to their source code write some additional code that would change the game behaviour as follows:

```
-bash-3.2$ javac Spiel.java
-bash-3.2$ java Spiel 3
Player 0: [3] false
Player 1: [2] false
Player 2: [2] false
Player 0: [3, 1] false
Player 1: [2, 1] false
Player 2: [2, 4] false
Player 0: [3, 1, 4] false
Player 1: [2, 1, 1] false
Player 2: [2, 4, 3] false
Player 0: [3, 1, 4, 1] false
Player 1: [2, 1, 1, 5] false
Player 2: [2, 4, 3, 5] false
Player 0: [3, 1, 4, 1, 5] false
Player 1: [2, 1, 1, 5, 1] false
Player 2: [2, 4, 3, 5, 2] false
Player 0: [3, 1, 4, 1, 5, 3] false
Player 1: [2, 1, 1, 5, 1, 2] false
Player 2: [2, 4, 3, 5, 2, 6] true
```

This time the game doesn’t end as soon as one player becomes a winner. We wait until all players have taken turns and at the end of the round we identify all winners. You can tell that you need to define a class Spiel².java which would work with Game.java, Player.java and Dice.java to produce this modified game. Because the player code used is the one in the original Player.java file the criterion for being a winner is the same as in the beginning: score 21 or more points. In the example above Player 2 managed to be the first to score 21 or more points (22 points, to be more precise) and since the other two players have 17 and 12 points respectively Player 2 is the only winner.

² Spiel means Game in German.
More than one player can be a winner at the end of the round, since the game is completely random. Here’s a session in which Player 0 and Player 2 tie:

```
-bash-3.2$ java Spiel 3
Player 0: [4] false
Player 1: [3] false
Player 2: [6] false
Player 0: [4, 3] false
Player 1: [3, 1] false
Player 2: [6, 3] false
Player 0: [4, 3, 4] false
Player 1: [3, 1, 6] false
Player 2: [6, 3, 1] false
Player 0: [4, 3, 4, 2] false
Player 1: [3, 1, 6, 3] false
Player 2: [6, 3, 1, 3] false
Player 0: [4, 3, 4, 2, 2] false
Player 1: [3, 1, 6, 3, 1] false
Player 2: [6, 3, 1, 3, 6] false
Player 0: [4, 3, 4, 2, 2, 6] true
Player 1: [3, 1, 6, 3, 1, 3] false
Player 2: [6, 3, 1, 3, 6, 6] true
```

And here’s a round where everybody wins:

```
-bash-3.2$ java Spiel 3
Player 0: [2] false
Player 1: [2] false
Player 2: [1] false
Player 0: [2, 2] false
Player 1: [2, 1] false
Player 2: [1, 3] false
Player 0: [2, 2, 5] false
Player 1: [2, 1, 4] false
Player 2: [1, 3, 4] false
Player 0: [2, 2, 5, 5] false
Player 1: [2, 1, 4, 6] false
Player 2: [1, 3, 4, 5] false
Player 0: [2, 2, 5, 5, 4] false
Player 1: [2, 1, 4, 6, 4] false
Player 2: [1, 3, 4, 5, 6] false
Player 0: [2, 2, 5, 5, 4, 4] true
Player 1: [2, 1, 4, 6, 4, 5] true
Player 2: [1, 3, 4, 5, 6, 6] true
```

Of course you can run the game for any number of players, not just 3.
Define your Spiel.java or reconstitute mine below:

```java
import java.util.*;

class Spiel {

    public static void main(String[] args) {
        ArrayList<Player> players = new ArrayList<Player>();
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            players.add(new Player("Player " + i));
        Game a = new Spiel(players);
        a.start();
    }
}
```

Obviously the main method in Spiel is exactly the same as in Game.

This is a final exam based on Exam Five.

I thought we had the least amount of practice on that one.

On the rest of the material we had plenty of sample exercises and study guides. Please review those too.

I include the answers below but please make an effort to solve this exam on your own first.

Spend at least 5 minutes on each of the 5 problems mentioned above.
I include the answers on the next few pages, one class per page.

Please give the questions a fair shot before you look at the answers.

Thank you.
In what follows each class should be placed in its own file as shown.
import java.util.*;

class Player extends ArrayList<Integer> {
    String name;
    Player(String name) {
        this.name = name;
    }
    public String toString() {
        return this.name + " : " + super.toString() + " " + isWinner();
    }
    int score() {
        int sum = 0;
        for (Integer i : this)
            sum += i;
        return sum;
    }
    boolean isWinner() {
        return score() >= 21;
    }
    public static void main(String[] args) {
        Player a = new Player("Larry Bird");
        System.out.println(a);
        a.add(3);
        System.out.println(a);
        a.add(12);
        System.out.println(a);
        a.add(9);
        System.out.println(a);
    }
}
class Dice {
    int shake() {
        return (int) (Math.random() * 6 + 1);
    }
    public static void main(String[] args) {
        Dice d = new Dice();
        for (int i = 0; i < 10; i++)
            System.out.print(d.shake() + " ");
    }
}
import java.util.*;

class Game {
    ArrayList<Player> players;
    Dice d = new Dice();
    Game(ArrayList<Player> players) {
        this.players = players;
    }
    void start() {
        while (true) {
            for (Player p : players) {
                p.add(d.shake());
                System.out.println(p);
                if (p.isWinner())
                    return;
            }
        }
    }
    public static void main(String[] args) {
        ArrayList<Player> players = new ArrayList<Player>();
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            players.add(new Player("Player " + i));
        Game a = new Game(players);
        a.start();
    }
}
class Joueur extends Player {
    Joueur(String name) {
        super(name);
    }
    boolean isWinner() {
        return score() % 7 == 0;
    }
}
import java.util.*;

class Jeu {
    public static void main(String[] args) {
        ArrayList<Player> players = new ArrayList<Player>();
        int n = Integer.parseInt(args[0]);
        for (int i = 0; i < n; i++)
            players.add(new Joueur("Player " + i));
        Game a = new Game(players);
        a.start();
    }
}
import java.util.*;

class Spiel extends Game {
    Spiel(ArrayList<Player> players) {
        super(players);
    }
    void start() {
        boolean gameOn = true;
        while (gameOn) {
            for (Player p : players) {
                p.add(d.shake());
                if (p.isWinner())
                    gameOn = false;
                System.out.println(p);
            }
        }
    }
}

public static void main(String[] args) {
    ArrayList<Player> players = new ArrayList<Player>();
    int n = Integer.parseInt(args[0]);
    for (int i = 0; i < n; i++)
        players.add(new Player("Player " + i));
    Game a = new Spiel(players);
    a.start();
}