1. Create a project “ExceptionDemo” and copy all the files from lab folders to the src folder of your project. You can watch videos – How to do Practices from the course “**Java-Practice-Guideline**” available on the learning portal.

2. Modify the code in the `League` class so that more teams are requested than can currently be produced from the pool of players.
   a. Modify the parameters passed to the `createTeams` method in the `main` method of `League` so that four teams are created with eleven players in each.

```
Team[] theTeams =
   theLeague.createTeams
      ("The Robins,The Crows,The Swallows,The Owls", 11);
```

   b. Run the project. What happened? Even though the project built successfully, an `Exception` was raised.

```
java.lang.IndexOutOfBoundsException: Index: 0, Size: 0
...at java.util.ArrayList.rangeCheck(ArrayList.java:638)
...at java.util.ArrayList.get(ArrayList.java:414)
...at utility.PlayerDatabase.getTeam(PlayerDatabase.java:36)
...at soccer.League.createTeams(League.java:57)
...at soccer.League.main(League.java:31)
```

   This `java.lang.IndexOutOfBoundsException` is a `RuntimeException` and, therefore, unchecked, meaning that you did not have to check for it in your code that called the `getTeam` method of `PlayerDatabase`.

3. Write a ***try/catch*** block in the `main` method of `League` to catch this `Exception`.
   a. After the instantiation of `League`, and before the call to the `createTeams` method, add a `try` statement and an opening left brace.

```
try {
```

   b. Because the entire remainder of the `main` method is dependent on `theTeams` array being populated, place the closing right brace of the try block after the last method call in the `main` method.

```
<… code omitted …>
theLeague.showBestTeam(theTeams);
} // Closing brace for try block here
```

   c. Indent the code within the try block to make it more readable.

d. Add a ***catch*** block after the closing brace of the ***try*** block. For now, just catch an `Exception`.

```
catch (Exception e) {
}
```

e. Now you will use the `Exception` reference, `e`, to print out a stack trace. Look in Javadocs for the `Exception` method that prints a stack trace—`printStackTrace` looks promising! Notice that it requires a `PrintStream` to be passed to it. Where can you find a `PrintStream`? How about `System.out` or, better yet, `System.err`!

f. Add a line to print the stack trace of the `Exception` to the error console.

```
e.printStackTrace(System.err);
```

g. Run the application. Notice that NetBeans prints this to the output window, but in red to indicate that it is using the `PrintStream` in `System.err`.

4. This is all very well, but it is just giving us the default behavior we get if we allow the unchecked `Exception` to pass right up through the call stack. Now we could print out a message telling the user that an `IndexOutOfBoundsException` has been raised and omit the stack trace, but wouldn't it be much better if the user gets a message telling them that there is a specific problem with the `PlayerDatabase` class?

5. Create a custom `Exception` to report the exact problem with `PlayerDatabase`.

a. Create a class named `PlayerDatabaseException` in the `utility` package that extends `Exception` (and therefore it must be caught).

```
package utility;
public class PlayerDatabaseException extends Exception {
}
```

b. Add a no parameter constructor, and also add a constructor that takes a `String` and calls the constructor on `Exception` (its superclass), passing the `String message`.

```
public PlayerDatabaseException() {}
public PlayerDatabaseException(String message) {
    super(message);
}
```

6. Add code to the `createTeams` method of `PlayerDatabase` to throw the `PlayerDatabaseException` when there is a problem. There are two approaches.

- Try to figure out if the problem will occur. For example, analyze the number of teams requested and the size of each and throw a `PlayerDatabaseException` if there are not enough players available.
- Check for an `IndexOutOfBoundsException` and, if caught, rethrow as a `PlayerDatabaseException`.

We will use the second approach here.

a. Look at the `getTeam` method of `PlayerDatabase` of the `utility` package. The problem we witnessed earlier is caused when no players remain in the `ArrayList`.

b. Add a `try {` immediately before the line that tries to get a `Player` reference from the `ArrayList`, and put the closing brace after the `players.remove()` method call. The try block will now look like this:

```
try {
   teamPlayers[i] = players.get(playerIndex);
   players.remove(playerIndex);
}
```

c. Add a `catch` block that checks for an `IndexOutOfBoundsException` and, if one is caught, rethrows a new `PlayerDatabaseException` with a suitable message (you will need to import `utility.PlayerDatabaseException`).

```
catch (IndexOutOfBoundsException ie) {
   throw new PlayerDatabaseException("Not enough players in the
database for the teams requested.");
}
```

d. Notice that NetBeans shows an error. The `PlayerDatabaseException` is an `Exception` (not a `RuntimeException`) and thus it must be caught. However, because you have already written ***try/catch*** code in `League`, all you need do here is throw the `PlayerDatabaseException` on up the stack. Modify the `getTeam` method signature as shown below.

```
public Player[] getTeam(int numberOfPlayers) throws
PlayerDatabaseException {
```

e. Now you will see another error, this time in `League`. You now need to modify the signature of the `createTeams` method in `League` in the same way so that it throws `PlayerDatabaseException` (you will also have to import `PlayerDatabaseException`).

f. Run the project again. Now the `Exception` will be caught in the `main` method of `League` and will display the following message.

```
utility.PlayerDatabaseException: Not enough players in the
database for the teams requested.
   at utility.PlayerDatabase.getTeam(PlayerDatabase.java:39)
   at soccer.League.createTeams(League.java:57)
   at soccer.League.main(League.java:31)
```