DELIVERABLES:

1. A pdf file containing all the source code, execution results, window captures and SQL*PLUS or SQL

Developer screen captures showing the database tables before and after modification. File name is: COP4703_lastName-FirstName_Assignemnt4.pdf

Source code:

1. TeamDriver.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

public class TeamDriver
{
	public static void main(String[] args) throws SQLException {

		String username = "aflorez2012";
		String password = "1037574336";

		// get a connection to DB
		Connection connection = getConnection(username , password); //1.a

		//print the table
		printTeamTable(connection); //1.b

		//declare variables
		Team team = new NBATeam( ); //1.c

		//modify some data
		System.out.println("Before add ..");
		team.printNumberOfTeams(connection); //1.d
		@SuppressWarnings("unused")
		int teamID_jazz = team.add(connection, "Jazz", "Utah"); //1.e
		@SuppressWarnings("unused")
		int teamID_raptors = team.add(connection, "Raptors", "Toronto"); //1.e
		System.out.println("After add ..");
		team.printNumberOfTeams(connection);


		team.updateCity(connection, "Clippers", "Los Angles"); //1.e
		team.updateName(connection, "Blazers", "Trail Blazers"); //1.e
		team.updateChampionships(connection, "Mavericks"); //1.e
		System.out.println("After update ..");
		printTeamTable(connection);
		team.remove(connection, teamID_jazz); //1.e
```

```
		//print the updated tables
		System.out.println("After remove ..");

		Collection TeamColl = getTeamCollectionArrayList(connection);
		printTeamCollection (TeamColl);

	}

	private static void printTeamCollection(Collection teamColl)
	{
		System.out.println("Team ID" + "\t" + "Team name" + "\t" + "Team city" +
"\t" + "Team Rank");
		System.out.println("-----------------------------------------------------
--------------------");
		if (teamColl instanceof Map)
		{
			Map theMap = (Map) teamColl;
			Set keys=theMap.keySet();
			Iterator keyIterator = keys.iterator();
			while(keyIterator.hasNext())
			{

	System.out.println(theMap.get(keyIterator.next()).toString());
			}
		}
		else
		{
			ArrayList<NBATeam> teamlist = new ArrayList<NBATeam>();
			Iterator iterator = teamColl.iterator();
			while(iterator.hasNext())
			{
				System.out.println(iterator.next().toString());
			}
		}
	}

	private static Collection getTeamCollectionArrayList(Connection connection) {

		try
		{
			ArrayList<NBATeam> teamlist = new ArrayList<NBATeam>();
			Statement st = connection.createStatement();
			ResultSet srs = st.executeQuery("SELECT TEAM_ID, TNAME, RANK,
CITY  FROM Team");
			while(srs.next())
			{
				NBATeam team = new NBATeam(srs.getInt("TEAM_ID"),
srs.getString("TNAME"), srs.getString("CITY"), srs.getInt("RANK"));
				teamlist.add(team);
			}
			st.close();
			srs.close();
			return teamlist;
		}
```

```
		catch (SQLException e) {

			e.printStackTrace();
		}

		return null;
	}

	private static void printTeamTable(Connection connection)
	{
		try
		{
			Statement statement = connection.createStatement();
			ResultSet resultset = statement.executeQuery("SELECT team_id,
tname, rank, city, championships  from TEAM");

			System.out.println("Team ID" + "\t" + "Team name" + "\t" + "Team
city" + "\t" + "Team rank" + "\t" + "Team Championships");
			System.out.println("-----------------------------------------
--------------------------");

			while (resultset.next())
			{
				int teamid = resultset.getInt(1);
				String teamname = resultset.getString(2);
				int teamrank = resultset.getInt(3);
				String teamcity= resultset.getString(4);
				int teamchamps = resultset.getInt(5);
				System.out.format("%2d%15s%15s%10d%15d", teamid, teamname,
teamcity, teamrank, teamchamps);
				System.out.print("\n");
			}
			resultset.close();
			statement.close();
			//connection.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}

	private static Connection getConnection(String username, String password)
	{
		try
		{
			System.out.println("\nGreeting a connection to Database.");
			Connection connection = null;
			connection = DriverManager.getConnection(
					"jdbc:oracle:thin:@131.91.168.91:1521:r11g",
username,password);

			if (connection == null)
			{
```

```
                        System.out.println("\n\nError: failed to obtain a
connection to the database, connection is null. Exiting..");
                        System.exit(1);
                    }
                    else
                    {
                        System.out.println("\nSuccessfully obtaining a connection
to the database...");
                        //connection.close();
                        return connection;
                    }
                }
                catch (SQLException e)
                {
                    System.out.println("Exception sql");
                    e.printStackTrace();
                }

                return null;
        }
}
```

Team.java

```
import java.sql.Connection;

public interface Team
{
        void getName();

        void getCity();

        void printNumberOfTeams(Connection connection);

        int add(Connection connection, String name, String city);

        void updateCity(Connection connection, String name, String city);

        void updateName(Connection connection, String oldName, String newName);

        void updateChampionships(Connection connection, String name);

        void remove(Connection connection, int teamID);
}
```

NBATeam.java

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
public class NBATeam implements Team {

	private int teamID = 0;
	private String teamName = null;
	private String teamCity = null;
	private int teamRank = 0;

	public NBATeam(int int1, String string, String string2, int int2)
	{
		this.teamID = int1;
		this.teamName = string;
		this.teamCity = string2;
		this.teamRank = int2;
	}

	public String toString()  //String method overloaded
	{
		return(getTeamID()+ "\t" + getTeamName()+ "\t" +getTeamCity() + "\t"
+getTeamRank());
	}

	public NBATeam() //default constructor
	{

	}

	NBATeam(NBATeam c) //copy constructor
	{

	}

	//setters and getters

	private int getTeamID() {
		return teamID;
	}

	private void setTeamID(int teamID) {
		this.teamID = teamID;
	}

	private String getTeamName() {
		return teamName;
	}

	private void setTeamName(String teamName) {
		this.teamName = teamName;
	}

	private String getTeamCity() {
		return teamCity;
	}

	private void setTeamCity(String teamCity) {
		this.teamCity = teamCity;
```

```
	}

	private int getTeamRank() {
		return teamRank;
	}

	private void setTeamRank(int teamRank) {
		this.teamRank = teamRank;
	}

	@Override
	public void getName()
	{
	}

	@Override
	public void getCity()
	{
	}

	public void printNumberOfTeams(Connection connection)
	{
		try
		{
			Statement statement = connection.createStatement();
			ResultSet resultset = statement.executeQuery("SELECT
count(team_id) from TEAM");

			while (resultset.next())
			{
				String team = resultset.getString(1);
				System.out.println("\nNumber of teams: " + team);
			}

			resultset.close();
			statement.close();
			//connection.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}

	@Override
	public int add(Connection connection, String name, String city)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := TEAM_pkg.add_team(?,?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setString(2, name);
			callablestatement.setString(3, city);
```

```
                        ResultSet result = callablestatement.executeQuery();
                        int retValue = callablestatement.getInt(1);
                        System.out.println("\nNew team ID: " + retValue + "\n" );
                        callablestatement.close();
                        result.close();
                        return retValue;
                }
                catch (SQLException e)
                {
                        e.printStackTrace();
                }
                return 0;
        }

        @Override
        public void updateCity(Connection connection, String name, String city)
        {
                try
                {
                        CallableStatement callablestatement =
connection.prepareCall("{call ? := TEAM_pkg.updatecity(?,?)}");
                        callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
                        callablestatement.setString(2, name);
                        callablestatement.setString(3, city);
                        ResultSet result = callablestatement.executeQuery();
                        int retValue = callablestatement.getInt(1);
                        callablestatement.close();
                        result.close();
                }
                catch (SQLException e)
                {
                        e.printStackTrace();
                }


        }

        @Override
        public void updateName(Connection connection, String oldName, String newName)
        {
                try
                {
                        CallableStatement callablestatement =
connection.prepareCall("{call ? := TEAM_pkg.updatename(?,?)}");
                        callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
                        callablestatement.setString(2, oldName);
                        callablestatement.setString(3, newName);
                        ResultSet result = callablestatement.executeQuery();
                        int retValue = callablestatement.getInt(1);
                        callablestatement.close();
                        result.close();
                }
                catch (SQLException e)
```

```
		{
			e.printStackTrace();
		}

	}

	@Override
	public void updateChampionships(Connection connection, String name)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := TEAM_pkg.updateChampionships(?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setString(2, name);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			callablestatement.close();
			result.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}

	@Override
	public void remove(Connection connection, int teamID)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := TEAM_pkg.remove(?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setInt(2, teamID);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			callablestatement.close();
			result.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}

	}

}
```

Execution results:

Before:

	TEAM_ID	TNAME	RANK	CITY	CHAMPIONSHIPS
1	1	Clippers	6	LA	0
2	2	Bulls	5	Chicago	6
3	3	Hornets	9	Charlotte	0
4	4	Blazers	2	Portland	0
5	5	Spurs	7	San Antonio	5
6	6	Mavericks	4	Dallas	3

```
// get a connection to DB
Connection connection = getConnection(username , password); //1.a
```

```
Greeting a connection to Database.

Successfully obtaining a connection to the database...
```

```
//print the table
printTeamTable(connection); //1.b
```

```
Team ID Team name       Team city       Team rank       Team Championships
--------------------------------------------------------------------------
 1       Clippers             LA         6               0
 2          Bulls        Chicago         5               6
 3        Hornets      Charlotte         9               0
 4        Blazers       Portland         2               0
 5          Spurs    San Antonio         7               5
 6      Mavericks         Dallas         4               3
```

```
//modify some data
System.out.println("Before add ..");
team.printNumberOfTeams(connection); //1.d
```

```
Before add ..

Number of teams: 6
```

```
int teamID_jazz = team.add(connection, "Jazz", "Utah"); //1.e
@SuppressWarnings("unused")
int teamID_raptors = team.add(connection, "Raptors", "Toronto"); //1.e
System.out.println("After add ..");
team.printNumberOfTeams(connection);
```

```
Before add ..

Number of teams: 6

New team ID: 1127


New team ID: 1128
```

	TEAM_ID	TNAME	RANK	CITY	CHAMPIONSHIPS
1	1	Clippers	6	LA	0
2	2	Bulls	5	Chicago	6
3	3	Hornets	9	Charlotte	0
4	4	Blazers	2	Portland	0
5	5	Spurs	7	San Antonio	5
6	6	Mavericks	4	Dallas	3
7	1127	Jazz	(null)	Utah	(null)
8	1128	Raptors	(null)	Toronto	(null)

```
System.out.println("After add ..");
team.printNumberOfTeams(connection);
```

```
After add ..

Number of teams: 8
```

```
team.updateCity(connection, "Clippers", "Los Angeles"); //1.e
team.updateName(connection, "Blazers", "Trail Blazers"); //1.e
team.updateChampionships(connection, "Mavericks"); //1.e
System.out.println("After update ..");
printTeamTable(connection);
```

```
After update ..
Team ID Team name       Team city       Team rank       Team Championships
------------------------------------------------------------------------
   1       Clippers     Los Angeles         6              0
   2          Bulls         Chicago         5              6
   3        Hornets       Charlotte         9              0
   4  Trail Blazers        Portland         2              0
   5          Spurs     San Antonio         7              5
   6      Mavericks          Dallas         4              4
1127           Jazz            Utah         0              0
1128        Raptors         Toronto         0              0
```

	TEAM_ID	TNAME	RANK	CITY	CHAMPIONSHIPS
1	1	Clippers	6	Los Angeles	0
2	2	Bulls	5	Chicago	6
3	3	Hornets	9	Charlotte	0
4	4	Trail Blazers	2	Portland	0
5	5	Spurs	7	San Antonio	5
6	6	Mavericks	4	Dallas	4
7	1127	Jazz	(null)	Utah	(null)
8	1128	Raptors	(null)	Toronto	(null)

```
team.remove(connection, teamID_jazz); //1.e
//print the updated tables
System.out.println("After remove ..");
Collection TeamColl = getTeamCollectionArrayList(connection);
printTeamCollection (TeamColl);
```

```
After remove ..
Team ID Team name       Team city       Team Rank
--------------------------------------------------------------------------------
   1       Clippers    Los Angeles             6
   2          Bulls        Chicago             5
   3        Hornets      Charlotte             9
   4  Trail Blazers       Portland             2
   5          Spurs    San Antonio             7
   6      Mavericks         Dallas             4
1128        Raptors        Toronto             0
```

	TEAM_ID	TNAME	RANK	CITY	CHAMPIONSHIPS
1	1	Clippers	6	Los Angeles	0
2	2	Bulls	5	Chicago	6
3	3	Hornets	9	Charlotte	0
4	4	Trail Blazers	2	Portland	0
5	5	Spurs	7	San Antonio	5
6	6	Mavericks	4	Dallas	4
7	1128	Raptors	(null)	Toronto	(null)

Source code:

2. PlayerDriver.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Collection;

public class PlayerDriver {

	//class variables
	private static Collection playerColl;

	public static void main(String[] args)
	{
		// get a connection to DB
		Connection connection = getConnection( );

		//declare local variables
		Player player = new NBAPlayer(); //2.b

		//print the table
		player.printTable(connection); //2.c

		//modify some data
		player.printNumberOfPlayers(connection); //2.d
		int playerID_hill =
		player.add(connection, "Grant Hill", "Small forward", 1994, "Duke");
//2.e
		int playerID_nash =
		player.add(connection,"Steve Nash","Point guard",1996,"Santa
Clara");//2.e
		player.printNumberOfPlayers(connection);
		player.printTable(connection);
		player.retire(connection, "Grant Hill", 2013); //2.e
		player.retire(connection, playerID_nash, 2015); //2.e

		//print the updated tables
		playerColl = player.getCollection(connection); //2.f
		player.printCollection(playerColl); //2.f
	}

	private static Connection getConnection()
	{
		try
		{
			System.out.println("\nGreeting a connection to Database.");
			Connection connection = null;
			connection = DriverManager.getConnection(
					"jdbc:oracle:thin:@131.91.168.91:1521:r11g",
"aflorez2012","1037574336");

			if (connection == null)
```

```
                    {
                            System.out.println("\n\nError: failed to obtain a
connection to the database, connection is null. Exiting..");
                            System.exit(1);
                    }
                    else
                    {
                            System.out.println("\nSuccessfully obtaining a connection
to the database...");
                            //connection.close();
                            return connection;
                    }
            }
            catch (SQLException e)
            {
                    System.out.println("Exception sql");
                    e.printStackTrace();
            }
            return null;
      }

}
```

Player.java

```
import java.sql.Connection;
import java.util.Collection;

public interface Player
{
      void printTable(Connection connection);

      void printNumberOfPlayers(Connection connection);

      int add(Connection connection, String playername, String playerposition, int
playerdraftyear,
                  String playereducation);

      void retire(Connection connection, String playername, int draftyear);

      void retire(Connection connection, int playerID, int draftyear);

      java.util.Collection getCollection(Connection connection);

      void printCollection(Collection playerColl);

}
```

NBAPlayer.java

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```java
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;


public class NBAPlayer implements Player
{
	private int PLAYER_ID = 0;
	private String PNAME = null;
	private String POSITION = null;
	private int DRAFT_YEAR = 0;
	private int RETIRE_YEAR = 0;
	private String EDUCATION = null;

	private int getPLAYER_ID()
	{
		return PLAYER_ID;
	}

	private void setPLAYER_ID(int pLAYER_ID)
	{
		PLAYER_ID = pLAYER_ID;
	}

	private String getPNAME()
	{
		return PNAME;
	}

	private void setPNAME(String pNAME)
	{
		PNAME = pNAME;
	}

	private String getPOSITION()
	{
		return POSITION;
	}

	private void setPOSITION(String pOSITION)
	{
		POSITION = pOSITION;
	}

	private int getDRAFT_YEAR()
	{
		return DRAFT_YEAR;
	}

	private void setDRAFT_YEAR(int dRAFT_YEAR)
	{
		DRAFT_YEAR = dRAFT_YEAR;
```

```
	}

	private int getRETIRE_YEAR()
	{
		return RETIRE_YEAR;
	}

	private void setRETIRE_YEAR(int rETIRE_YEAR)
	{
		RETIRE_YEAR = rETIRE_YEAR;
	}

	private String getEDUCATION()
	{
		return EDUCATION;
	}

	private void setEDUCATION(String eDUCATION)
	{
		EDUCATION = eDUCATION;
	}

	public String toString()
	{
		return(getPLAYER_ID() + "\t" + getPNAME() + "\t" + getPOSITION() + "\t"
+ getDRAFT_YEAR() +
				"\t" + getRETIRE_YEAR() + "\t" + getEDUCATION());
	}

	public NBAPlayer(int pid, String pname, String ppos, int pdraft,
			int pretire, String peducation)
	{
		this.PLAYER_ID = pid;
		this.PNAME = pname;
		this.POSITION = ppos;
		this.DRAFT_YEAR = pdraft;
		this.RETIRE_YEAR = pretire;
		this.EDUCATION = peducation;
	}

	public NBAPlayer()  //default constructor
	{

	}

	NBAPlayer(NBAPlayer c) //copy constructor
	{

	}

	@Override
	public void printTable(Connection connection)
	{
		try
		{
```

```
                    Statement statement = connection.createStatement();
                    ResultSet resultset = statement.executeQuery("SELECT player_id,
pname, position, draft_year, retire_year, education from player");

                    System.out.println("Player ID" + "\t" + "Player name" + "\t" +
"Position" + "\t" + "Draft Year" + "\t" + "Retire Year" + "\t" + "Education");
                    System.out.println("-----------------------------------------
---------");

                    while (resultset.next())
                    {
                           int playerid = resultset.getInt(1);
                           String playername = resultset.getString(2);
                           String position = resultset.getString(3);
                           int draftyear = resultset.getInt(4);
                           int retireyear = resultset.getInt(5);
                           String education = resultset.getString(5);
                           System.out.format("%2d%25s%15s%2d%6d%20s", playerid,
playername, position, draftyear, retireyear, education);
                           System.out.print("\n");
                    }
                    resultset.close();
                    statement.close();
                    //connection.close();
             }
             catch (SQLException e)
             {
                    e.printStackTrace();
             }

      }

      @Override
      public void printNumberOfPlayers(Connection connection)
      {
             try
             {
                    Statement statement = connection.createStatement();
                    ResultSet resultset = statement.executeQuery("SELECT
count(player_id) from PLAYER");

                    while (resultset.next())
                    {
                           String player = resultset.getString(1);
                           System.out.println("\nNumber of Arenas: " + player);
                    }

                    resultset.close();
                    statement.close();
                    //connection.close();
             }
             catch (SQLException e)
             {
                    e.printStackTrace();
             }
```

```
	}

	@Override
	public int add(Connection connection, String playername,
			String playerposition, int playerdraftyear, String
playereducation)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := PLAYER_pkg.add(?,?,?,?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setString(2, playername);
			callablestatement.setString(3, playerposition);
			callablestatement.setInt(4, playerdraftyear);
			callablestatement.setString(5, playereducation);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			System.out.println("\nNew player ID: " + retValue + "\n" );
			result.close();
			callablestatement.close();
			return retValue;
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
		return 0;
	}

	@Override
	public void retire(Connection connection, String playername, int draftyear)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := PLAYER_pkg.retire(?,?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setString(2, playername);
			callablestatement.setInt(3, draftyear);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			System.out.println("\nPlayer ID removed: " + retValue + "\n" );
			callablestatement.close();
			result.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}
```

```
	@Override
	public void retire(Connection connection, int playerID, int draftyear)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := PLAYER_pkg.retire(?,?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setInt(2, playerID);
			callablestatement.setInt(3, draftyear);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			System.out.println("\nPlayer ID removed: " + retValue + "\n" );
			callablestatement.close();
			result.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}

	}

	@Override
	public Collection getCollection(Connection connection)
	{
		try
		{
			ArrayList<NBAPlayer> playerlist = new ArrayList<NBAPlayer>();
			Statement st = connection.createStatement();
			ResultSet srs = st.executeQuery("SELECT PLAYER_ID, PNAME,
POSITION, DRAFT_YEAR, RETIRE_YEAR, EDUCATION FROM PLAYER");
			while(srs.next())
			{
				NBAPlayer player = new NBAPlayer(srs.getInt("PLAYER_ID"),
srs.getString("PNAME"), srs.getString("POSITION"), srs.getInt("DRAFT_YEAR"),
srs.getInt("RETIRE_YEAR"), srs.getString("EDUCATION"));
				playerlist.add(player);
			}
			st.close();
			srs.close();
			return playerlist;
		}
		catch (SQLException e) {

			e.printStackTrace();
		}
		return null;
	}

	@Override
	public void printCollection(Collection playerColl)
	{
```

```
            System.out.println("Player ID" + "\t" + "Player name" + "\t" +
"Position" + "\t" + "Draft year"
                                          + "\t" + "Retire year" + "\t" + "Education");
            System.out.println("-----------------------------------------------------
--------------------");
            if (playerColl instanceof Map)
            {
                  Map theMap = (Map) playerColl;
                  Set keys=theMap.keySet();
                  Iterator keyIterator = keys.iterator();
                  while(keyIterator.hasNext())
                  {

      System.out.println(theMap.get(keyIterator.next()).toString());
                  }
            }
            else
            {
                  ArrayList<NBAPlayer> playerlist = new ArrayList<NBAPlayer>();
                  Iterator iterator = playerColl.iterator();
                  while(iterator.hasNext())
                  {
                        System.out.println(iterator.next().toString());
                  }

            }

      }

}
```

Execution results:

Before:

	PLAYER_ID	PNAME	POSITION	DRAFT_YEAR	RETIRE_YEAR	EDUCATION
1	1	Chris Paul	Small Forward	2005	(null)	Wake Forest
2	2	Tony Parker	Point guard	2001	(null)	INSEP
3	3	Marco Belinelli	Shooting guard	2007	(null)	San Giovanni
4	4	Gary Neal	Power Forward	2007	(null)	La Salle University
5	5	Kawhi Leonard	Forward	2011	(null)	San Diego State
6	6	Patty Mills	Point guard	2009	(null)	Marist College
7	7	Tyson Chandler	Center	2001	(null)	Dominguez
8	8	Derek Fisher	Shooting guard	1996	2014	Arkansas

```
// get a connection to DB
Connection connection = getConnection( );
```

```
Greeting a connection to Database.

Successfully obtaining a connection to the database...
```

Player ID	Player name	Position	Draft Year	Retire Year	Education
1	Chris Paul	Small Forward	2005	0	null
2	Tony Parker	Point guard	2001	0	null
3	Marco Belinelli	Shooting guard	2007	0	null
4	Gary Neal	Power Forward	2007	0	null
5	Kawhi Leonard	Forward	2011	0	null
6	Patty Mills	Point guard	2009	0	null
7	Tyson Chandler	Center	2001	0	null
8	Derek Fisher	Shooting guard	1996	2014	2014

```
//modify some data
player.printNumberOfPlayers(connection); //2.d
```

```
Number of Players: 8
```

```
int playerID_hill =
      player.add(connection, "Grant Hill", "Small forward", 1994, "Duke"); //2.e
int playerID_nash =
      player.add(connection,"Steve Nash","Point guard",1996,"Santa Clara");//2.e
player.printNumberOfPlayers(connection);
```

```
New player ID: 9


New player ID: 10


Number of Players: 10
```

Player ID	Player name	Position	Draft Year	Retire Year	Education
1	Chris Paul	Small Forward	2005	0	null
2	Tony Parker	Point guard	2001	0	null
3	Marco Belinelli	Shooting guard	2007	0	null
4	Gary Neal	Power Forward	2007	0	null
5	Kawhi Leonard	Forward	2011	0	null
6	Patty Mills	Point guard	2009	0	null
7	Tyson Chandler	Center	2001	0	null
8	Derek Fisher	Shooting guard	1996	2014	2014
9	Grant Hill	Small forward	1994	0	null
10	Steve Nash	Point guard	1996	0	null

	PLAYER_ID	PNAME	POSITION	DRAFT_YEAR	RETIRE_YEAR	EDUCATION
1	1	Chris Paul	Small Forward	2005	(null)	Wake Forest
2	2	Tony Parker	Point guard	2001	(null)	INSEP
3	3	Marco Belinelli	Shooting guard	2007	(null)	San Giovanni
4	4	Gary Neal	Power Forward	2007	(null)	La Salle University
5	5	Kawhi Leonard	Forward	2011	(null)	San Diego State
6	6	Patty Mills	Point guard	2009	(null)	Marist College
7	7	Tyson Chandler	Center	2001	(null)	Dominguez
8	8	Derek Fisher	Shooting guard	1996	2014	Arkansas
9	9	Grant Hill	Small forward	1994	(null)	Duke
10	10	Steve Nash	Point guard	1996	(null)	Santa Clara

```
player.retire(connection, "Grant Hill", 2013); //2.e
player.retire(connection, playerID_nash, 2015); //2.e*/
```

```
 Player ID retire updated: 9


 Player ID retire updated: 10
```

```
//print the updated tables
playerColl = player.getCollection(connection); //2.f
player.printCollection(playerColl); //2.f
```

```
Player ID       Player name     Position        Draft year      Retire year     Education
-------------------------------------------------------------------------------------------
 1               Chris Paul  Small Forward          2005               0         Wake Forest
 2              Tony Parker    Point guard          2001               0               INSEP
 3          Marco Belinelli Shooting guard          2007               0        San Giovanni
 4                Gary Neal  Power Forward          2007               0 La Salle University
 5             Kawhi Leonard        Forward          2011               0     San Diego State
 6               Patty Mills    Point guard          2009               0      Marist College
 7            Tyson Chandler         Center          2001               0           Dominguez
 8              Derek Fisher Shooting guard          1996            2014            Arkansas
 9                Grant Hill  Small forward          1994            2013                Duke
10                Steve Nash    Point guard          1996            2015        Santa Clara
```

	PLAYER_ID	PNAME	POSITION	DRAFT_YEAR	RETIRE_YEAR	EDUCATION
1	1	Chris Paul	Small Forward	2005	(null)	Wake Forest
2	2	Tony Parker	Point guard	2001	(null)	INSEP
3	3	Marco Belinelli	Shooting guard	2007	(null)	San Giovanni
4	4	Gary Neal	Power Forward	2007	(null)	La Salle University
5	5	Kawhi Leonard	Forward	2011	(null)	San Diego State
6	6	Patty Mills	Point guard	2009	(null)	Marist College
7	7	Tyson Chandler	Center	2001	(null)	Dominguez
8	8	Derek Fisher	Shooting guard	1996	2014	Arkansas
9	9	Grant Hill	Small forward	1994	2013	Duke
10	10	Steve Nash	Point guard	1996	2015	Santa Clara

Source code:

3. ArenaDriver.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ArenaDriver {

	//class variables
	private static java.util.Collection<NBAArena> arenaColl;

	public static void main(String[] args)
	{
		// get a connection to DB
		Connection connection = getConnection( );

		//declare local variable
		Arena arena = new NBAArena(); // 3.b

		//print the table
		Arena.printTable(connection); // 3.c

		//modify some data
		arena.printNumberOfArenas(connection); // 3.d
		int arenaID_oracle = arena.add(connection, "Oracle Arena", "Oakland");
// 3.e
		int arenaID_pepsi = arena.add(connection, "Pepsi Center", "Denver"); //
3.e
		int arenaID_toyota = arena.add(connection, "Toyota Center", "Houston");
//3.e
		arena.printNumberOfArenas(connection);
		Arena.printTable(connection);
		arena.remove(connection, "Oracle Arena"); // 3.e
		arena.remove(connection, arenaID_pepsi); // 3.e
		arena.remove(connection, arenaID_toyota); // 3.e

		//print the updated tables
		arenaColl = Arena.getCollection(connection); // 3.f
		arena.printCollection(arenaColl); // 3.f*/
	}

	private static Connection getConnection()
	{
		try
		{
			System.out.println("\nGreeting a connection to Database.");
			Connection connection = null;
			connection = DriverManager.getConnection(
					"jdbc:oracle:thin:@131.91.168.91:1521:r11g",
"aflorez2012","1037574336");
			if (connection == null)
			{
```

```
                        System.out.println("\n\nError: failed to obtain a
connection to the database, connection is null. Exiting..");
                        System.exit(1);
                    }
                    else
                    {
                        System.out.println("\nSuccessfully obtaining a connection
to the database...");
                        //connection.close();
                        return connection;
                    }
                }
                catch (SQLException e)
                {
                    System.out.println("Exception sql");
                    e.printStackTrace();
                }
                return null;
        }
}
```

Arena.java

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;


public abstract class Arena
{
    static void printTable(Connection connection)
    {
        try
        {
            Statement statement = connection.createStatement();
            ResultSet resultset = statement.executeQuery("SELECT arena_id,
aname, city from Arena");

            System.out.println("Arena ID" + "\t" + "Arena name" + "\t" +
"Arena city");
            System.out.println("-----------------------------------------
---------");

            while (resultset.next())
            {
                int arenaid = resultset.getInt(1);
                String arenaname = resultset.getString(2);
                String arenacity = resultset.getString(3);
                System.out.format("%2d%25s%15s", arenaid, arenaname,
arenacity);
                System.out.print("\n");
            }
```

```
                        resultset.close();
                        statement.close();
                        //connection.close();
                }
                catch (SQLException e)
                {
                        e.printStackTrace();
                }
        }

        abstract void printNumberOfArenas(Connection connection);

        abstract int add(Connection connection, String string, String string2);

        abstract void remove(Connection connection, String arenaname);

        abstract void remove(Connection connection, int arenaid);

        static Collection<NBAArena> getCollection(Connection connection)
        {
                try
                {
                        ArrayList<NBAArena> arenalist = new ArrayList<NBAArena>();
                        Statement st = connection.createStatement();
                        ResultSet srs = st.executeQuery("SELECT ARENA_ID, ANAME, CITY
FROM Arena");
                        while(srs.next())
                        {
                                //System.out.print("\n" + srs.getInt("TEAM_ID"));
                                NBAArena arena = new NBAArena(srs.getInt("ARENA_ID"),
srs.getString("ANAME"), srs.getString("CITY"));
                                arenalist.add(arena);
                        }
                        st.close();
                        srs.close();
                        return arenalist;
                }
                catch (SQLException e) {

                        e.printStackTrace();
                }
                return null;
        }

        abstract void printCollection(Collection<NBAArena> arenaColl);

}
```

NBAArena.java

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```java
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;


public class NBAArena extends Arena
{
	private int ARENAID = 0;
	private String ARENANAME = null;
	private String ARENACITY = null;

	private int getARENAID() {
		return ARENAID;
	}

	private void setARENAID(int aRENAID) {
		ARENAID = aRENAID;
	}

	private String getARENANAME() {
		return ARENANAME;
	}

	private void setARENANAME(String aRENANAME) {
		ARENANAME = aRENANAME;
	}

	private String getARENACITY() {
		return ARENACITY;
	}

	private void setARENACITY(String aRENACITY) {
		ARENACITY = aRENACITY;
	}

	public NBAArena(int ARENA_ID, String ARENA_NAME, String ARENA_CITY)
	{
		this.ARENAID = ARENA_ID;
		this.ARENANAME = ARENA_NAME;
		this.ARENACITY = ARENA_CITY;
	}

	public NBAArena()  //default constructor
	{

	}

	NBAArena(NBAArena c) //copy constructor
	{

	}
```

```
	public String toString()
	{
		return String.format("%2d%25s%15s", getARENAID(), getARENANAME(),
getARENACITY());
	}

	@Override
	void printNumberOfArenas(Connection connection)
	{
		try
		{
			Statement statement = connection.createStatement();
			ResultSet resultset = statement.executeQuery("SELECT
count(arena_id) from ARENA");

			while (resultset.next())
			{
				String arena = resultset.getString(1);
				System.out.println("\nNumber of Arenas: " + arena);
			}

			resultset.close();
			//connection.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}

	@Override
	int add(Connection connection, String arenaname, String arenacity)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := ARENA_pkg.add(?,?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setString(2, arenaname);
			callablestatement.setString(3, arenacity);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			System.out.println("\nNew Arena ID: " + retValue + "\n" );
			callablestatement.close();
			result.close();
			return retValue;

		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
		return 0;
	}
```

```
	@Override
	void remove(Connection connection, String arenaname)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := ARENA_pkg.remove(?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setString(2, arenaname);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			System.out.println("\nArena ID removed: " + retValue + "\n" );
			callablestatement.close();
			result.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}
	}

	@Override
	void remove(Connection connection, int arenaid)
	{
		try
		{
			CallableStatement callablestatement =
connection.prepareCall("{call ? := ARENA_pkg.remove(?)}");
			callablestatement.registerOutParameter(1,
java.sql.Types.INTEGER);
			callablestatement.setInt(2, arenaid);
			ResultSet result = callablestatement.executeQuery();
			int retValue = callablestatement.getInt(1);
			System.out.println("\nNew Arena ID: " + retValue + "\n" );
			callablestatement.close();
			result.close();
		}
		catch (SQLException e)
		{
			e.printStackTrace();
		}

	}

	@Override
	void printCollection(Collection<NBAArena> arenaColl)
	{
		System.out.println("Arena ID" + "\t" + "Arena name" + "\t" + "Arena
city");
		System.out.println("-----------------------------------------------------
--------------------");
		if (arenaColl instanceof Map)
		{
```

```
                    Map theMap = (Map) arenaColl;
                    Set keys=theMap.keySet();
                    Iterator keyIterator = keys.iterator();
                    while(keyIterator.hasNext())
                    {

        System.out.println(theMap.get(keyIterator.next()).toString());
                    }
              }
              else
              {
                    ArrayList<NBAArena> arenalist = new ArrayList<NBAArena>();
                    Iterator iterator = arenaColl.iterator();
                    while(iterator.hasNext())
                    {
                          System.out.println(iterator.next().toString());
                    }
              }
        }
}
```

Execution results:

Before:

	ARENA_ID	ANAME	CITY
1	1	American Airlines	Miami
2	2	Moda Center	Portland
3	3	Staples Center	LA
4	4	United Center	Chicago
5	5	TD Garden	Boston
6	6	ATT Center	San Antonio
7	7	Philips Arena	Atlanta

```
// get a connection to DB
Connection connection = getConnection( ); // 3.a

 Greeting a connection to Database.

 Successfully obtaining a connection to the database...

//print the table
Arena.printTable(connection); // 3.c
```

```
Arena ID        Arena name       Arena city
-------------------------------------------------------
 1         American Airlines          Miami
 2               Moda Center       Portland
 3            Staples Center             LA
 4             United Center        Chicago
 5                 TD Garden         Boston
 6                ATT Center    San Antonio
 7             Philips Arena        Atlanta
```

	ARENA_ID	ANAME	CITY
1	1	American Airlines	Miami
2	2	Moda Center	Portland
3	3	Staples Center	LA
4	4	United Center	Chicago
5	5	TD Garden	Boston
6	6	ATT Center	San Antonio
7	7	Philips Arena	Atlanta

```
//modify some data
arena.printNumberOfArenas(connection); // 3.d
```

```
Number of Arenas: 7
```

```
int arenaID_oracle = arena.add(connection, "Oracle Arena", "Oakland"); // 3.e
int arenaID_pepsi = arena.add(connection, "Pepsi Center", "Denver"); // 3.e
int arenaID_toyota = arena.add(connection, "Toyota Center", "Houston"); //3.e
arena.printNumberOfArenas(connection);
```

```
New Arena ID: 8

New Arena ID: 9

New Arena ID: 10

Number of Arenas: 10
```

```
Arena.printTable(connection);
```

Arena ID	Arena name	Arena city
8	Oracle Arena	Oakland
9	Pepsi Center	Denver
10	Toyota Center	Houston
1	American Airlines	Miami
2	Moda Center	Portland
3	Staples Center	LA
4	United Center	Chicago
5	TD Garden	Boston
6	ATT Center	San Antonio
7	Philips Arena	Atlanta

```
arena.remove(connection, "Oracle Arena"); // 3.e
arena.remove(connection, arenaID_pepsi); // 3.e
arena.remove(connection, arenaID_toyota); // 3.e

//print the updated tables
arenaColl = Arena.getCollection(connection); // 3.f
arena.printCollection(arenaColl); // 3.f*/
```

Arena ID	Arena name	Arena city
1	American Airlines	Miami
2	Moda Center	Portland
3	Staples Center	LA
4	United Center	Chicago
5	TD Garden	Boston
6	ATT Center	San Antonio
7	Philips Arena	Atlanta