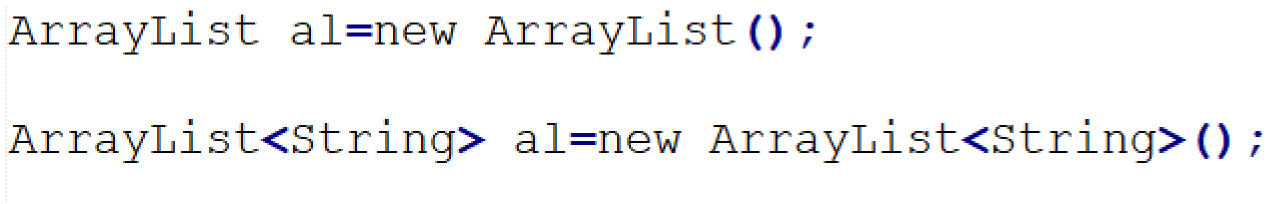
Java

Agenda

- ArrayList
- HashMap
- JDBC

ArrayList

- ArrayList is pre defined class in Java used for dynamic array for storing elements.
- ArrayList can contains duplicate elements.
- We can add, insert and remove elements from ArrayList.

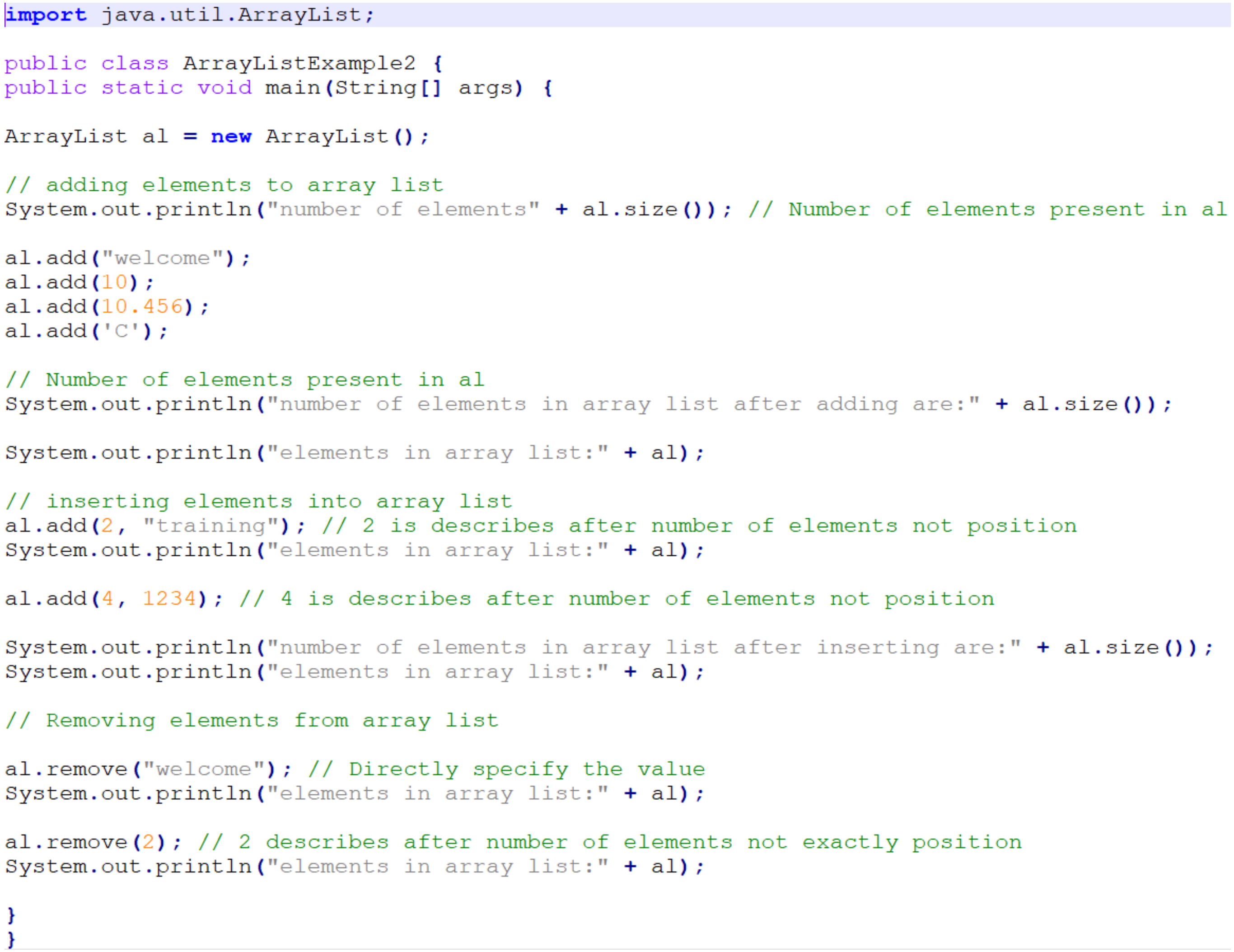


```
ArrayList al=new ArrayList();

ArrayList<String> al=new ArrayList<String>();
```

Java ArrayList Example1

```java
import java.util.ArrayList;
public class ArrayListExample {

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<String>();

// adding elements to array list
list.add("Raj");
list.add("Ravi");
list.add("Pavan");
list.add("Simran");
list.add("Arvinder");

System.out.println(list.size());    // returns number of elements in array list

for (String s : list) // reading elements from array list
{
System.out.println(s);
}
}
}
```

Java ArrayList Example2

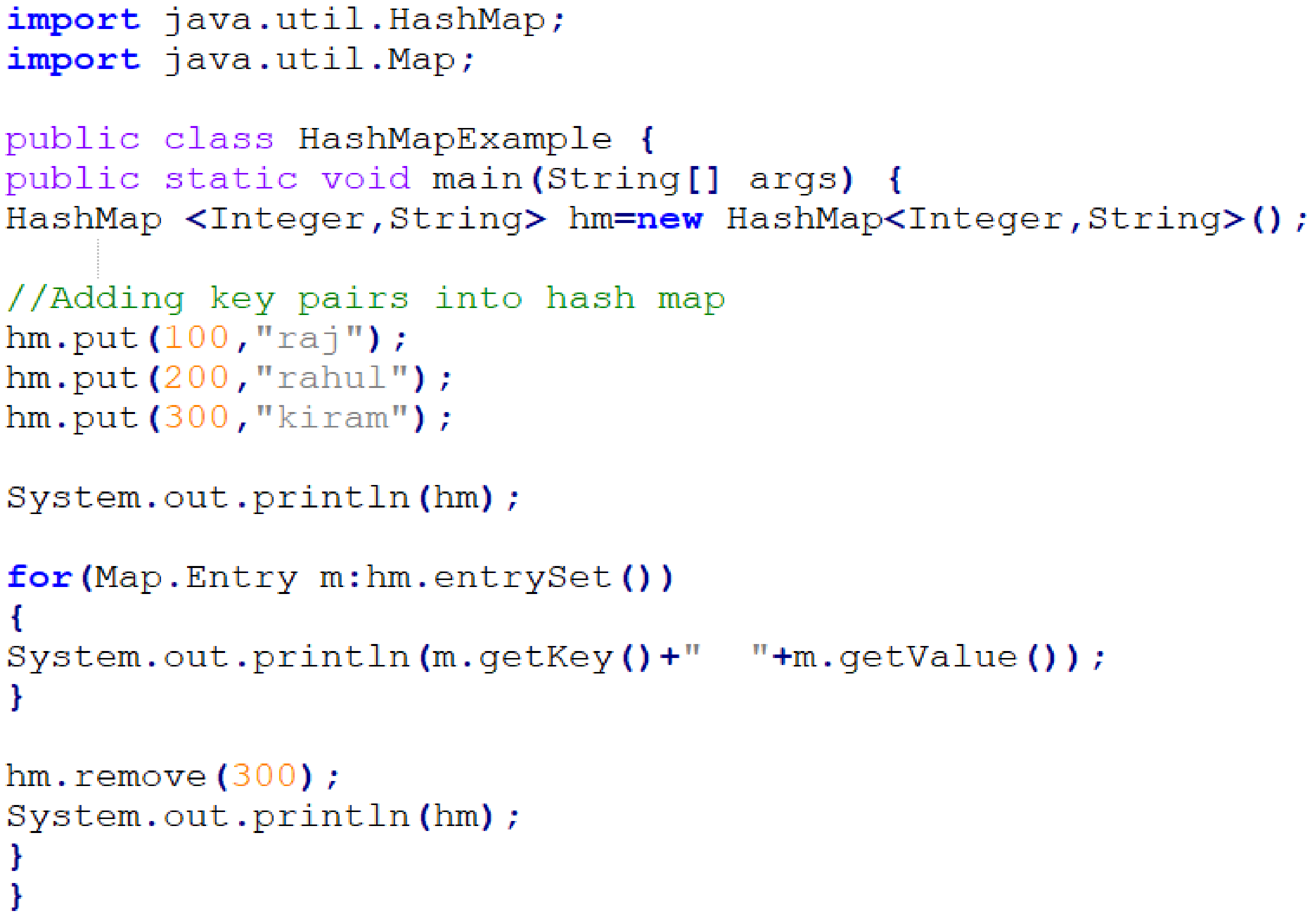


```
import java.util.ArrayList;

public class ArrayListExample2 {
public static void main(String[] args) {

ArrayList al = new ArrayList();

// adding elements to array list
System.out.println("number of elements" + al.size()); // Number of elements present in al

al.add("welcome");
al.add(10);
al.add(10.456);
al.add('C');

// Number of elements present in al
System.out.println("number of elements in array list after adding are:" + al.size());

System.out.println("elements in array list:" + al);

// inserting elements into array list
al.add(2, "training"); // 2 is describes after number of elements not position
System.out.println("elements in array list:" + al);

al.add(4, 1234); // 4 is describes after number of elements not position

System.out.println("number of elements in array list after inserting are:" + al.size());
System.out.println("elements in array list:" + al);

// Removing elements from array list

al.remove("welcome"); // Directly specify the value
System.out.println("elements in array list:" + al);

al.remove(2); // 2 describes after number of elements not exactly position
System.out.println("elements in array list:" + al);

}
}
```

HashMap

- The important points about Java HashMap:
  - A HashMap contains values based on the key.
  - It contains only unique elements.
  - It maintains no order.

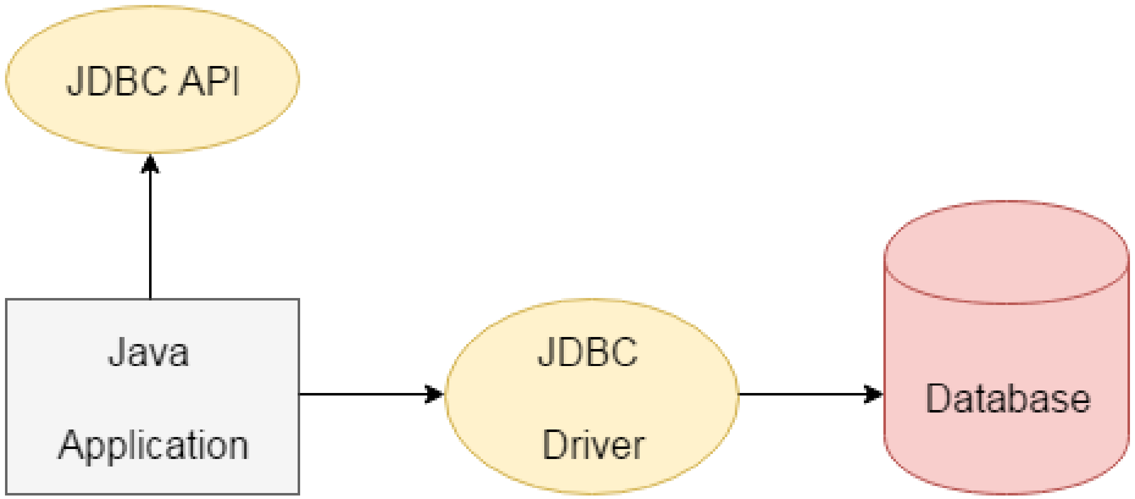
Java HashMap Example



```java
import java.util.HashMap;
import java.util.Map;

public class HashMapExample {
public static void main(String[] args) {
HashMap <Integer,String> hm=new HashMap<Integer,String>();

//Adding key pairs into hash map
hm.put(100,"raj");
hm.put(200,"rahul");
hm.put(300,"kiram");

System.out.println(hm);

for(Map.Entry m:hm.entrySet())
{
System.out.println(m.getKey()+"   "+m.getValue());
}

hm.remove(300);
System.out.println(hm);
}
}
```

JDBC – Java Database Connectivity

- Java JDBC is a java API to connect and execute query with the database.
- JDBC API uses jdbc drivers to connect with the database.



Database and SQL

- Database: stores the data in the tables.
- SQL- a language used for communicate to the database.
  - DML : Data Manipulation Language
  - DDL : Data Definition Language
  - DCL : Data Control Language
  - TCL : Transaction Language

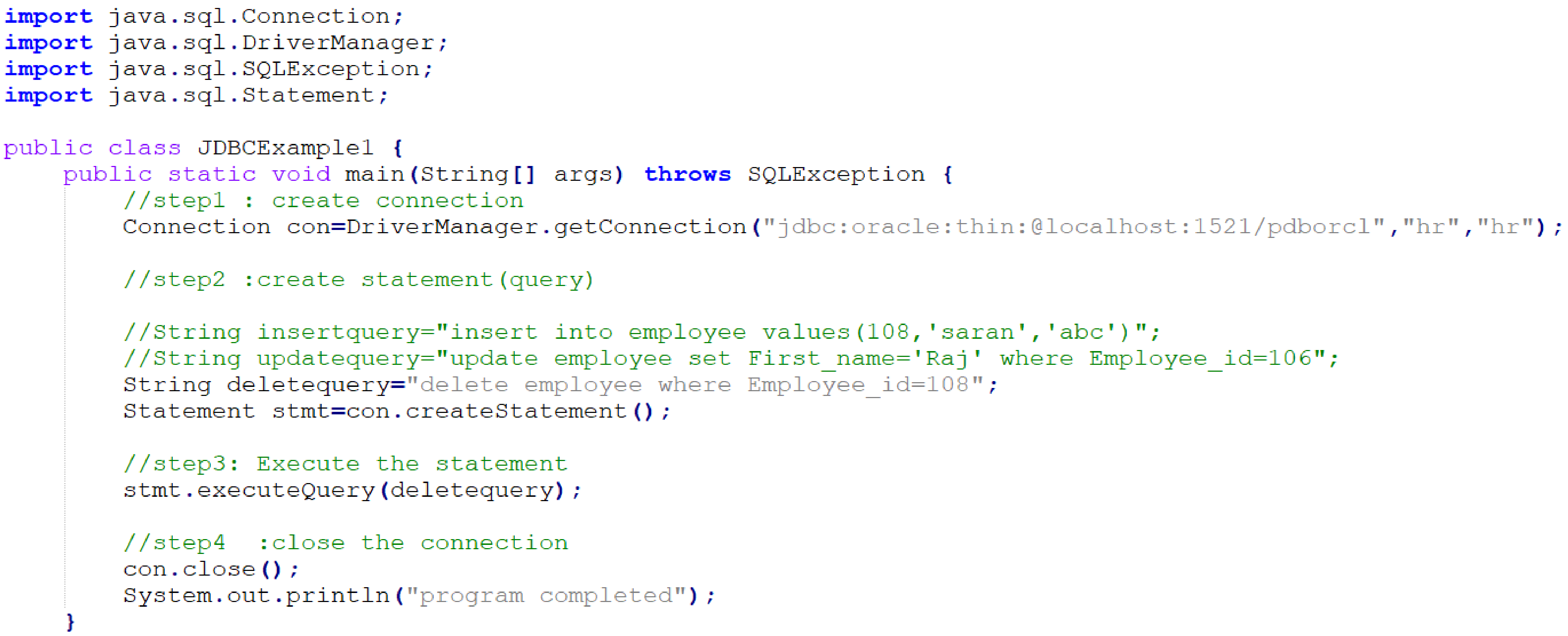
Database Components

- Database Client
  - CLI
  - GUI
- Database Server

4 Steps to connect to the database in java

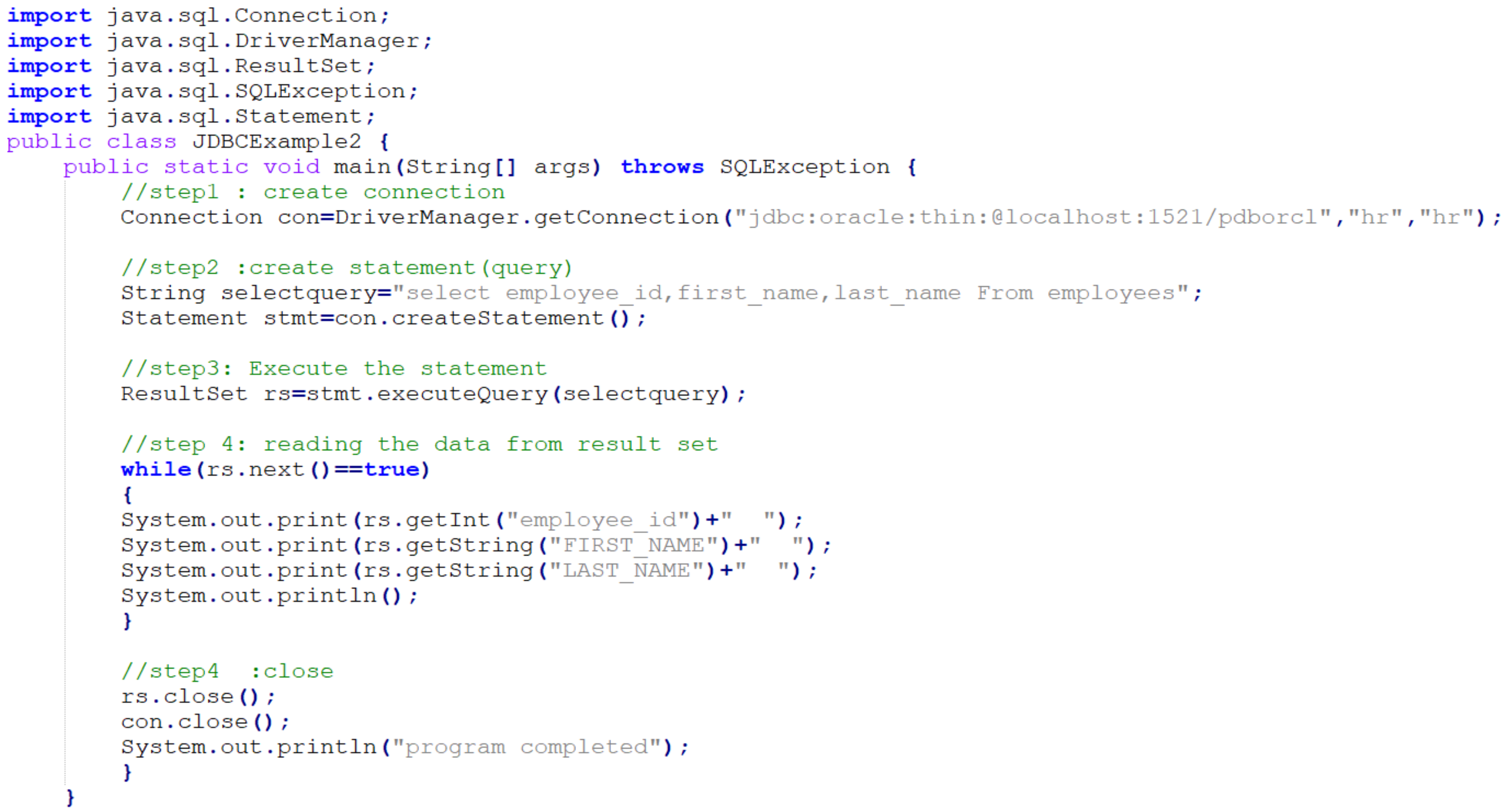
- Creating connection
- Creating statement
- Executing queries
- Closing connection

JDBC Example1



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

public class JDBCExample1 {
    public static void main(String[] args) throws SQLException {
        //step1 : create connection
        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521/pdborcl","hr","hr");

        //step2 :create statement(query)

        //String insertquery="insert into employee values(108,'saran','abc')";
        //String updatequery="update employee set First_name='Raj' where Employee_id=106";
        String deletequery="delete employee where Employee_id=108";
        Statement stmt=con.createStatement();

        //step3: Execute the statement
        stmt.executeQuery(deletequery);

        //step4  :close the connection
        con.close();
        System.out.println("program completed");
    }
```

JDBC Example2



```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class JDBCExample2 {
    public static void main(String[] args) throws SQLException {
        //step1 : create connection
        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521/pdborcl","hr","hr");

        //step2 :create statement(query)
        String selectquery="select employee_id,first_name,last_name From employees";
        Statement stmt=con.createStatement();

        //step3: Execute the statement
        ResultSet rs=stmt.executeQuery(selectquery);

        //step 4: reading the data from result set
        while(rs.next()==true)
        {
        System.out.print(rs.getInt("employee_id")+"   ");
        System.out.print(rs.getString("FIRST_NAME")+"   ");
        System.out.print(rs.getString("LAST_NAME")+"   ");
        System.out.println();
        }

        //step4  :close
        rs.close();
        con.close();
        System.out.println("program completed");
        }
    }
```