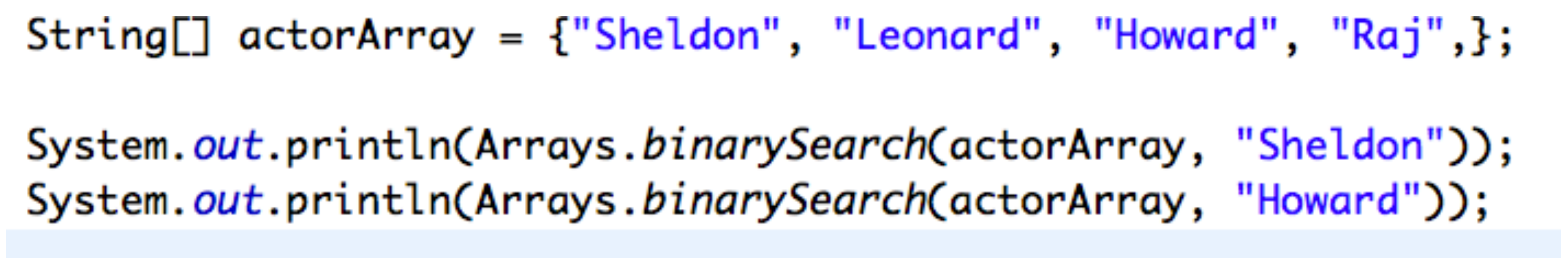
Java Time Machine

Wang Yang

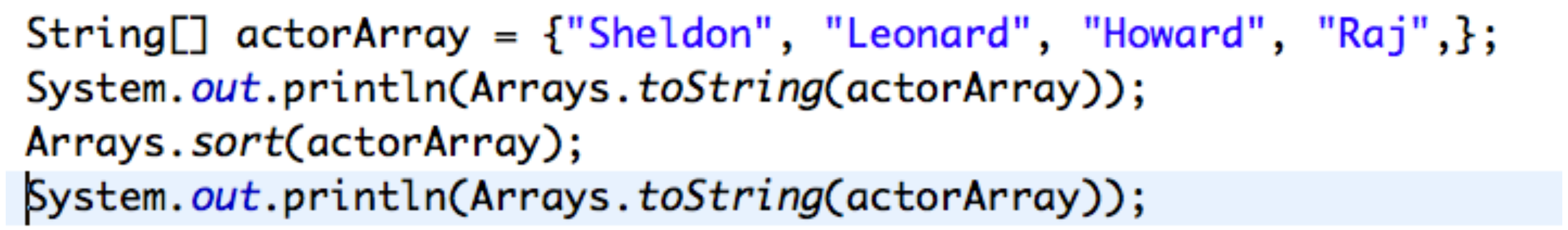
Arrays



```
String[] actorArray = {"Sheldon", "Leonard", "Howard", "Raj",};

System.out.println(Arrays.binarySearch(actorArray, "Sheldon"));
System.out.println(Arrays.binarySearch(actorArray, "Howard"));
```

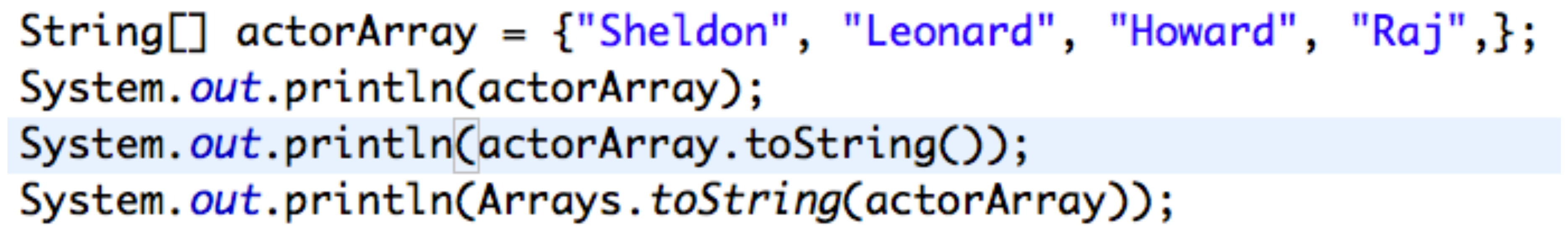
binarySearch only work for sorted array !!



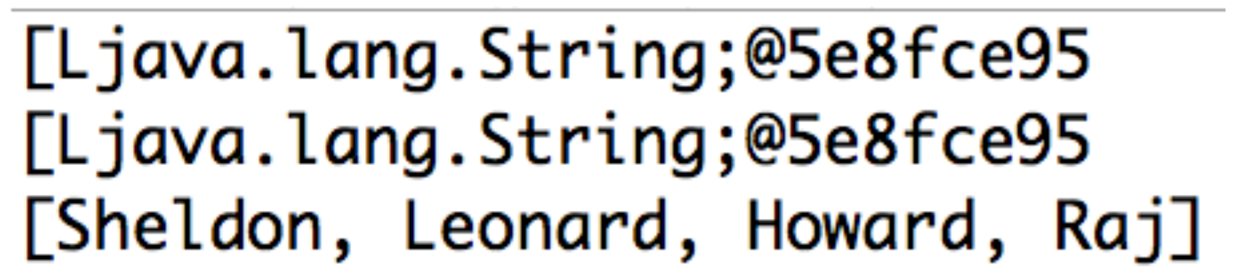
```
String[] actorArray = {"Sheldon", "Leonard", "Howard", "Raj",};
System.out.println(Arrays.toString(actorArray));
Arrays.sort(actorArray);
System.out.println(Arrays.toString(actorArray));
```

Arrays

- How to print the content of the Array



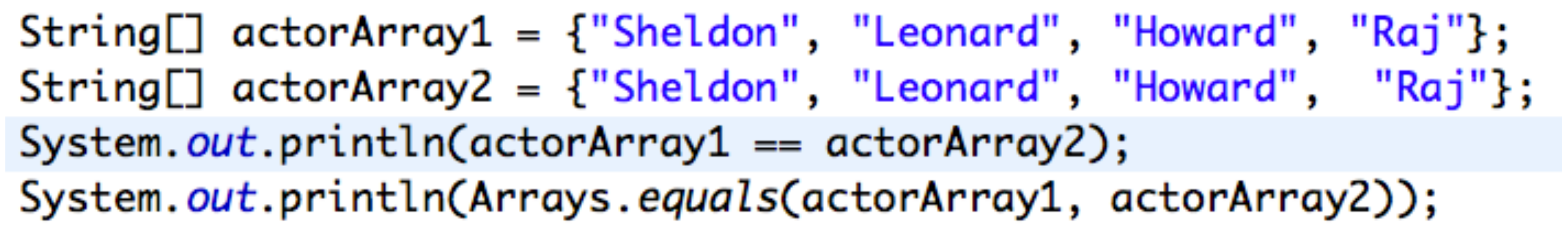
```java
String[] actorArray = {"Sheldon", "Leonard", "Howard", "Raj",};
System.out.println(actorArray);
System.out.println(actorArray.toString());
System.out.println(Arrays.toString(actorArray));
```



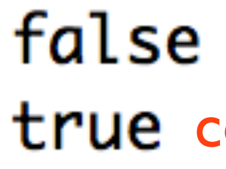
```
[Ljava.lang.String;@5e8fce95
[Ljava.lang.String;@5e8fce95
[Sheldon, Leonard, Howard, Raj]
```

Arrays

- How to compare whether two arrays are equal?



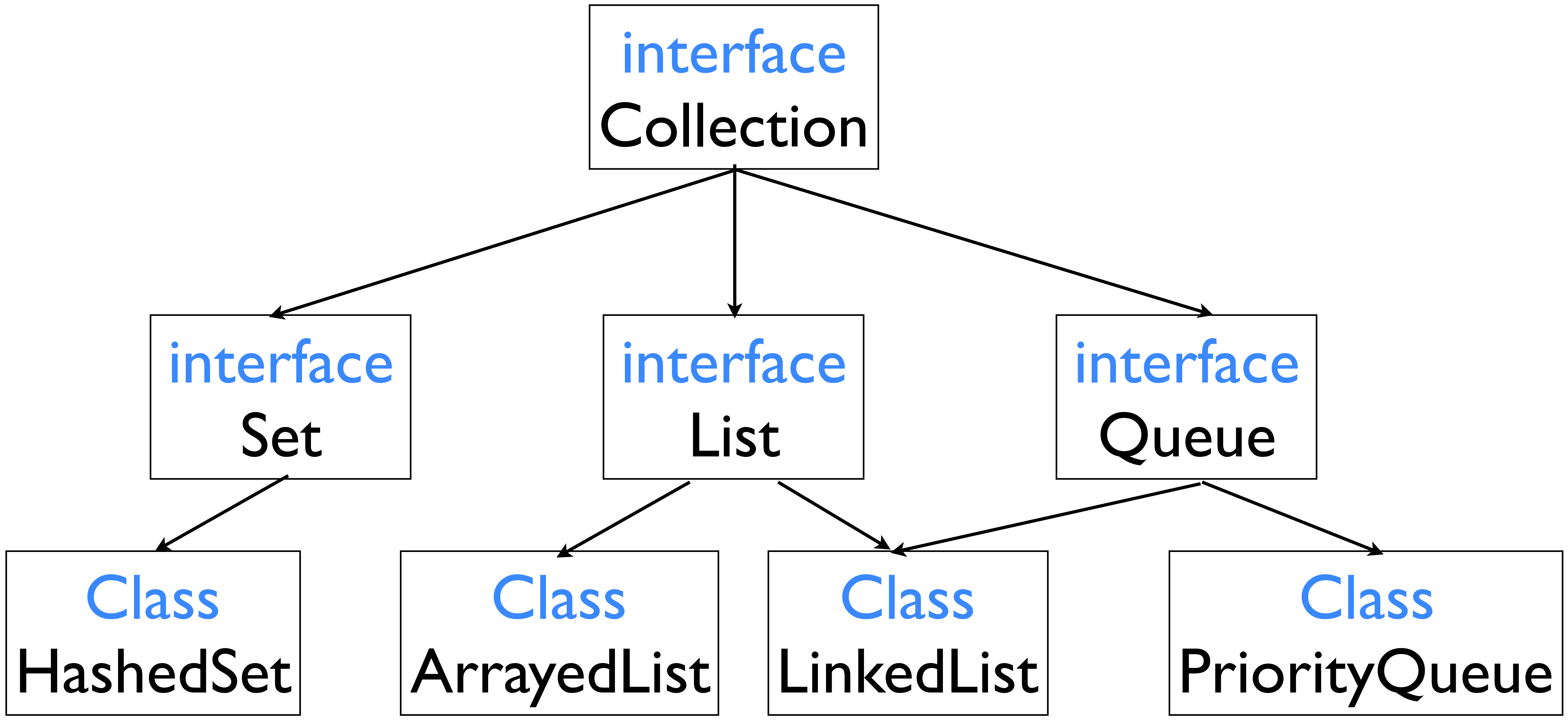
```
String[] actorArray1 = {"Sheldon", "Leonard", "Howard", "Raj"};
String[] actorArray2 = {"Sheldon", "Leonard", "Howard",  "Raj"};
System.out.println(actorArray1 == actorArray2);
System.out.println(Arrays.equals(actorArray1, actorArray2));
```



`false` refer_1 != refer_2

`true` content_1 == content_2

Collection

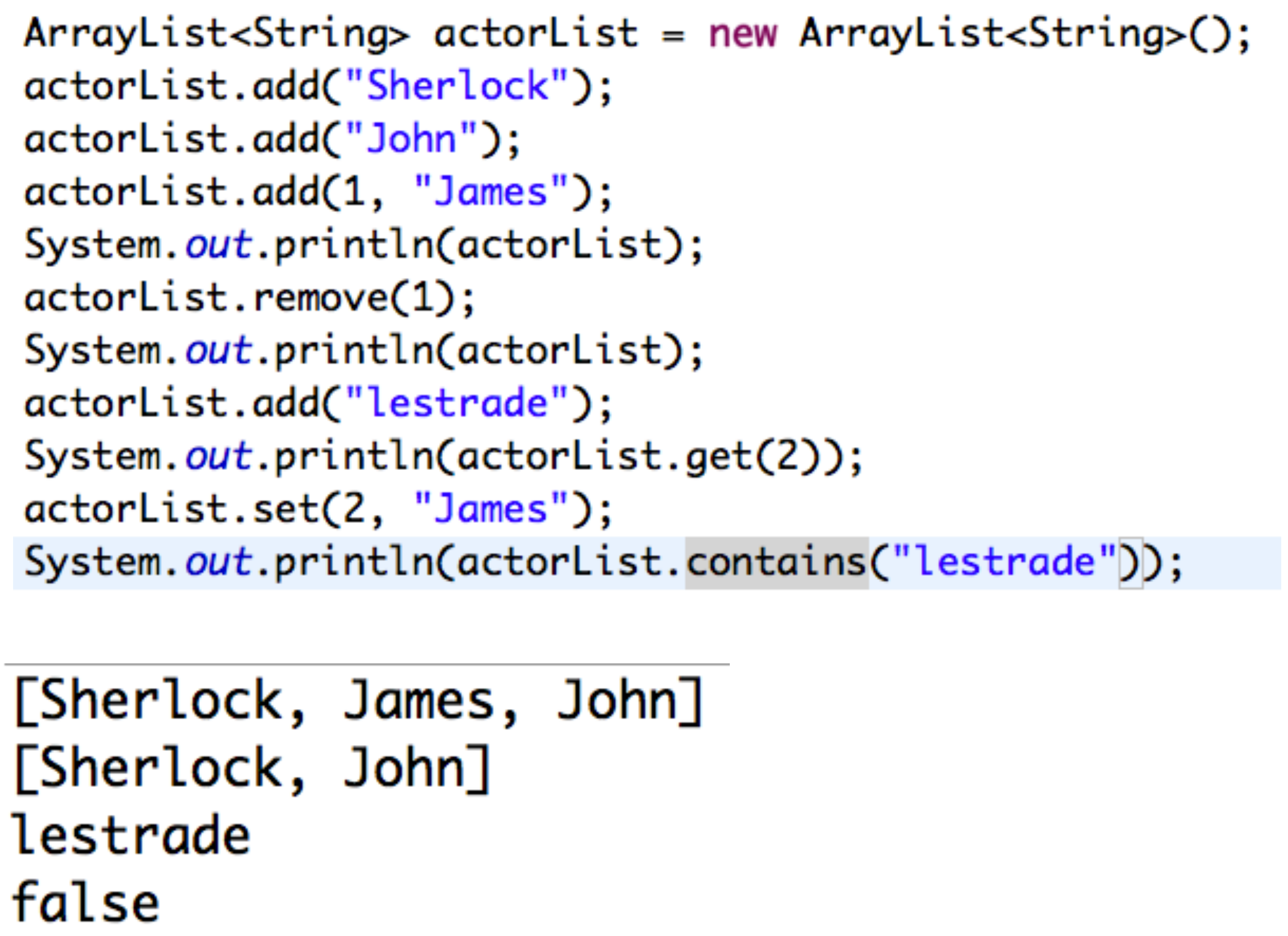


interface Collection
interface Set
interface List
interface Queue
Class HashedSet
Class ArrayedList
Class LinkedList
Class PriorityQueue

List

- List the method of List Interface
- add/remove/clear/contain/size
- toArray

List



```java
ArrayList<String> actorList = new ArrayList<String>();
actorList.add("Sherlock");
actorList.add("John");
actorList.add(1, "James");
System.out.println(actorList);
actorList.remove(1);
System.out.println(actorList);
actorList.add("lestrade");
System.out.println(actorList.get(2));
actorList.set(2, "James");
System.out.println(actorList.contains("lestrade"));
```

```
[Sherlock, James, John]
[Sherlock, John]
lestrade
false
```

List

- What’s the difference between ArrayList and LinkedList

- Array vs Link
- Different Interface,Different Method
- Different performance

List

- What’s the performance difference between ArrayList and LinkedList

- ArrayList:
  - Efficient in random access of elements
  - May enlarge backend array when append new elements (can be partly solved by setting initial capacity)
  - Not efficient for insertion (may cause the movement of elements)
  - Waste of space (solved by trimToSize)
- LinkedList
  - Do not cause the reassignment of memory
  - Efficient for add / delete / insert
  - Not efficient for random access (need traverse from head) difference between ArrayList and LinkedList

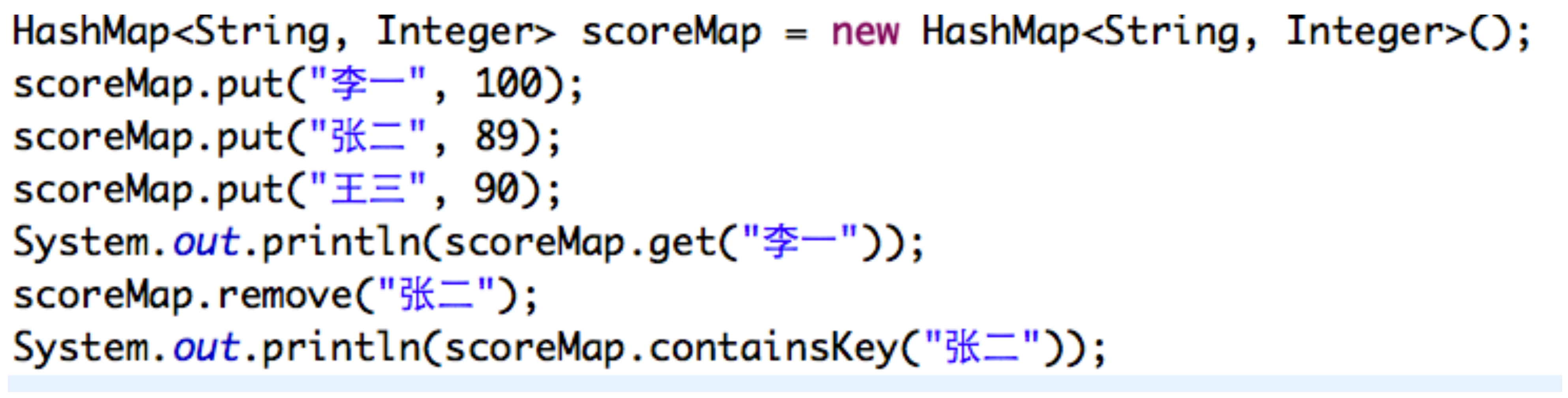
HashMap

- What’s the requirement of Key,Value in HashMap
- must be object

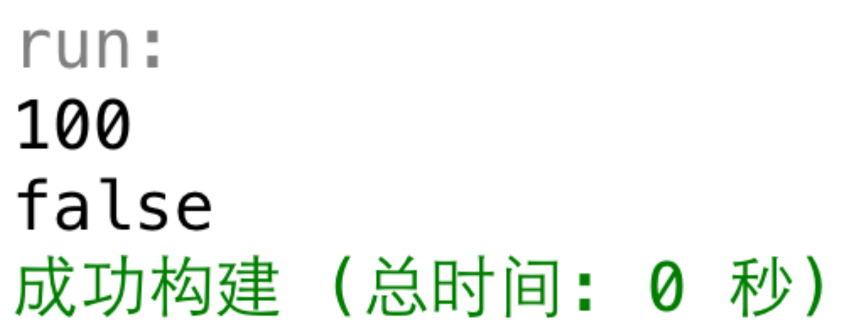
- cannot contain duplicate keys

- each key can map to at most one value

HashMap



```java
HashMap<String, Integer> scoreMap = new HashMap<String, Integer>();
scoreMap.put("李一", 100);
scoreMap.put("张二", 89);
scoreMap.put("王三", 90);
System.out.println(scoreMap.get("李一"));
scoreMap.remove("张二");
System.out.println(scoreMap.containsKey("张二"));
```



```
run:
100
false
成功构建 (总时间: 0 秒)
```

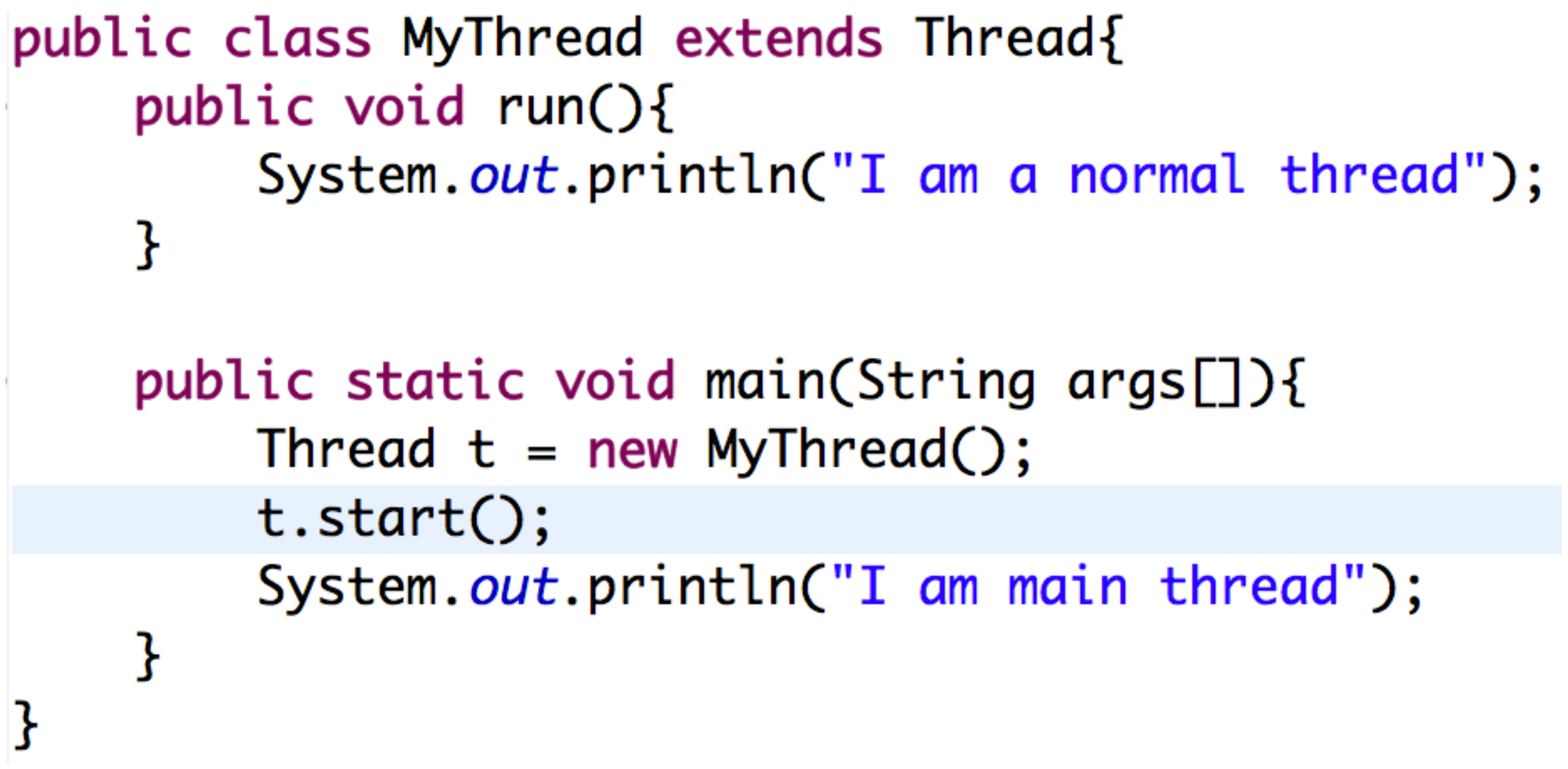
iterator

- list the method of iterator

- next()
- hasNext()

Thread

- Use Thread Class to define a thread and run it

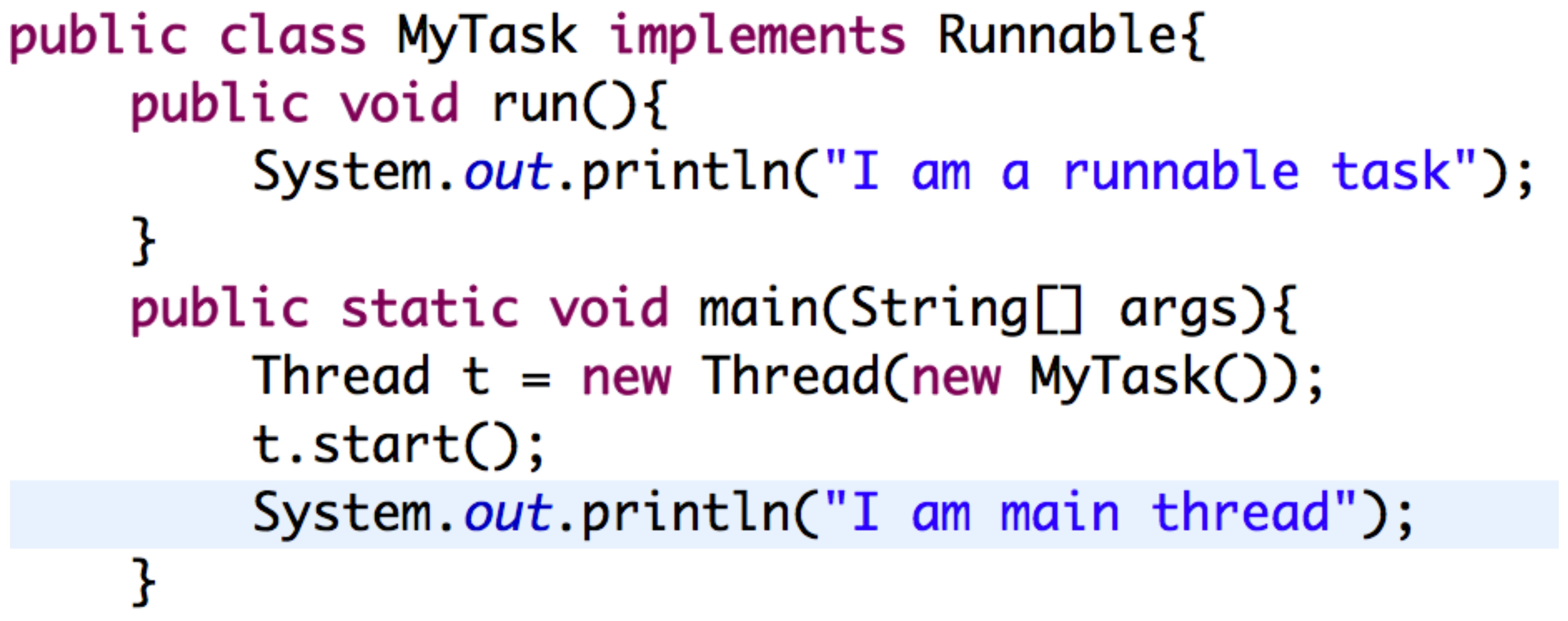


```java
public class MyThread extends Thread{
    public void run(){
        System.out.println("I am a normal thread");
    }

    public static void main(String args[]){
        Thread t = new MyThread();
        t.start();
        System.out.println("I am main thread");
    }
}
```

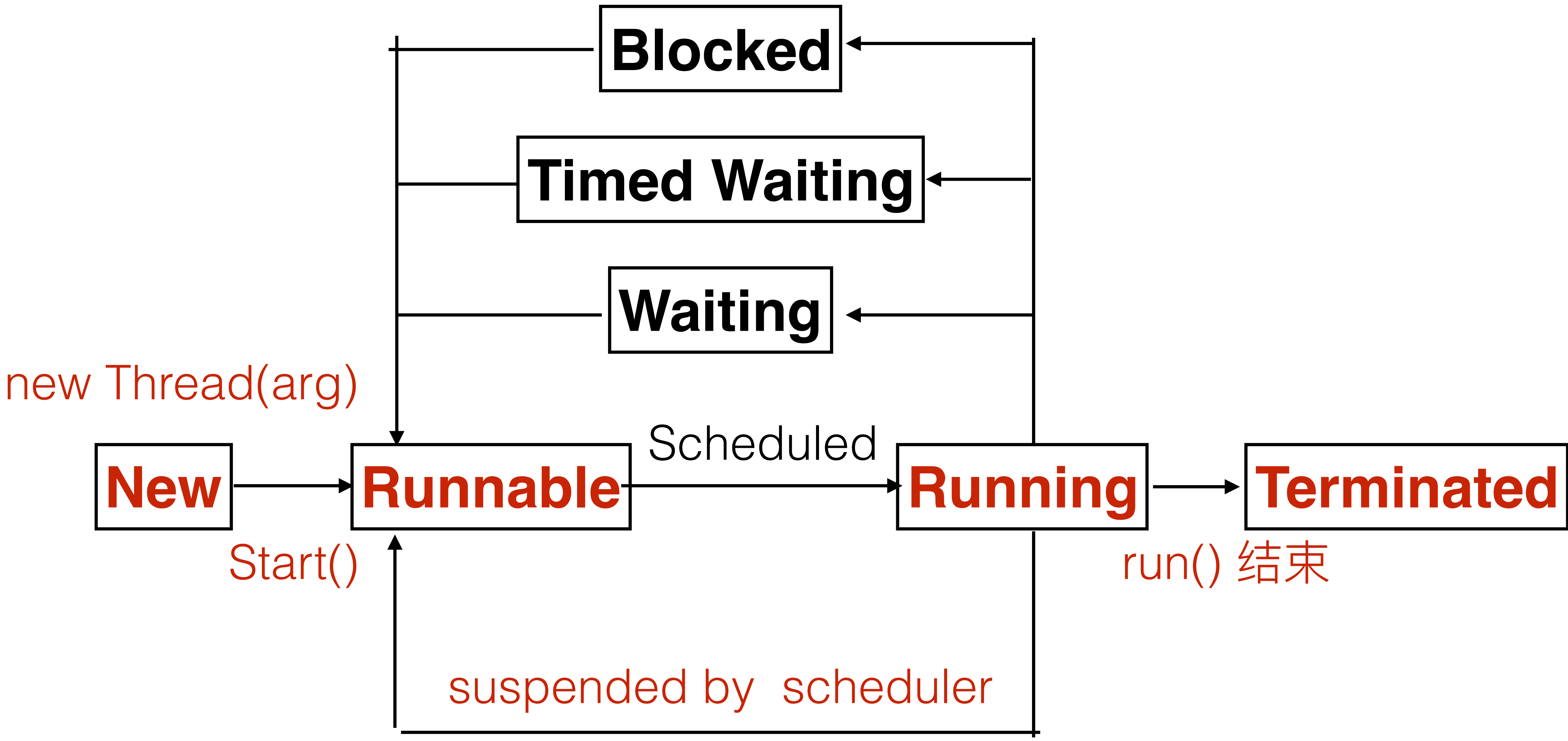
Thread

- Use Runnable Interface to define a thread and run it

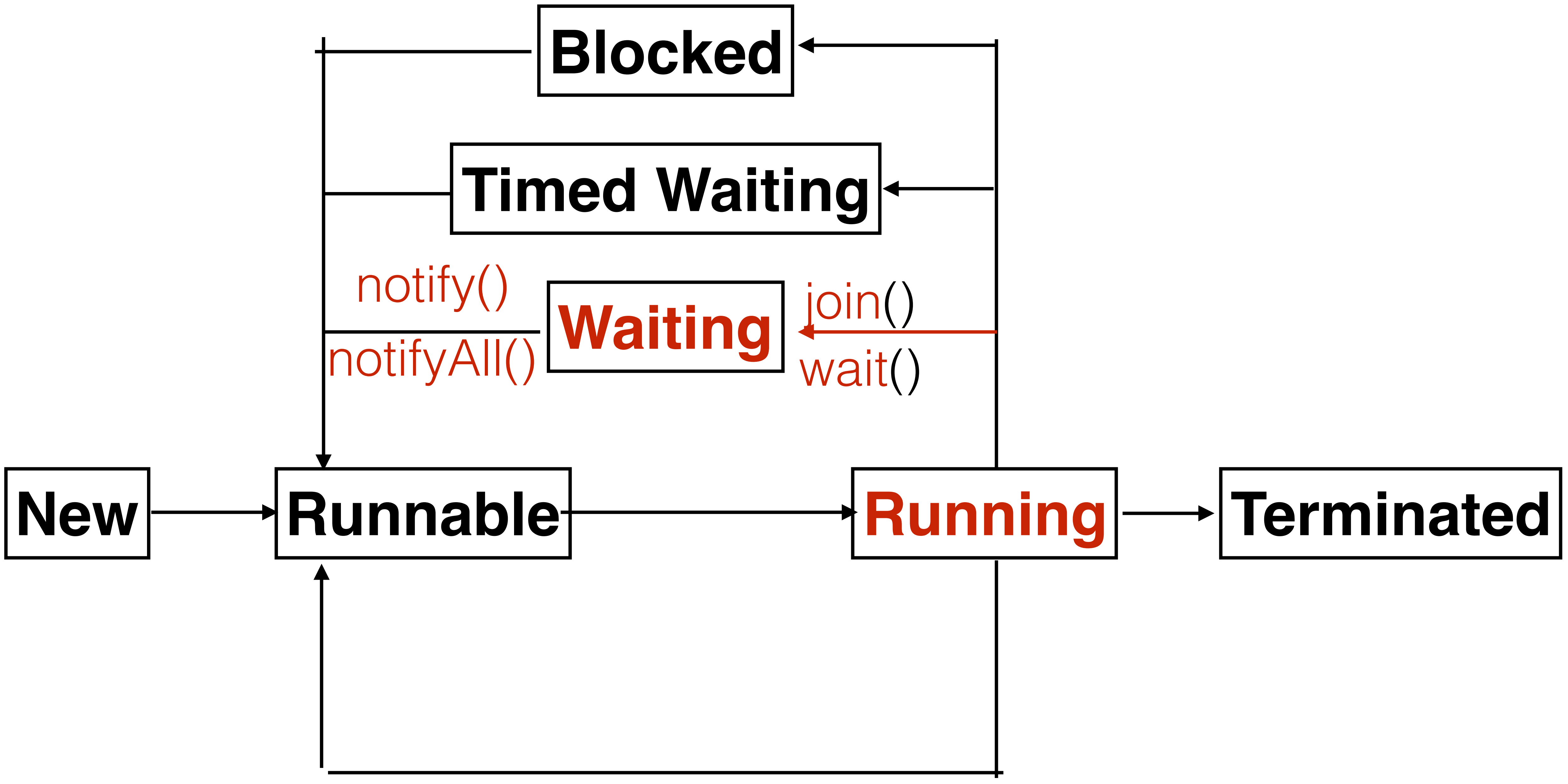


```java
public class MyTask implements Runnable{
    public void run(){
        System.out.println("I am a runnable task");
    }
    public static void main(String[] args){
        Thread t = new Thread(new MyTask());
        t.start();
        System.out.println("I am main thread");
    }
```

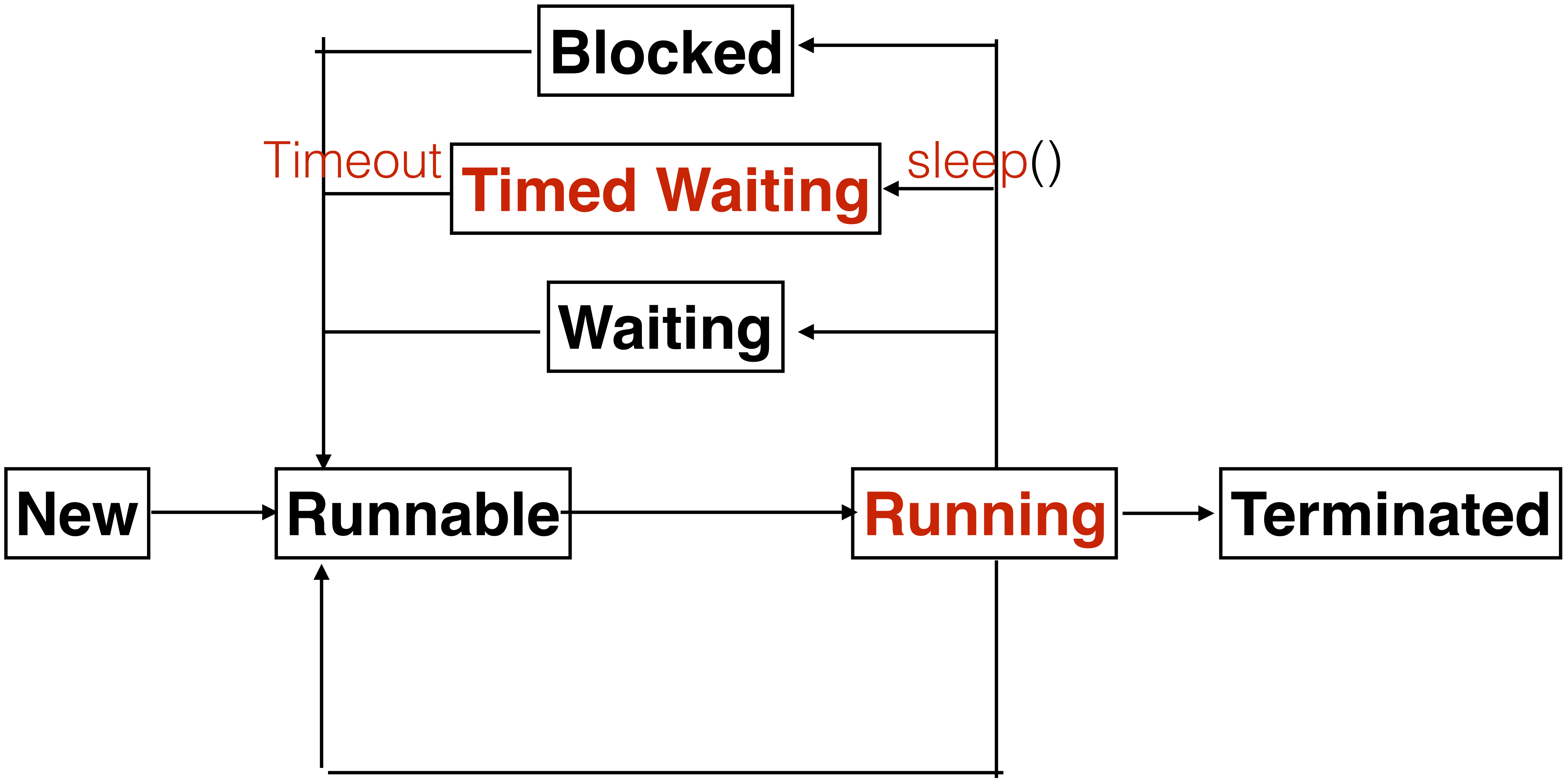
Thread State



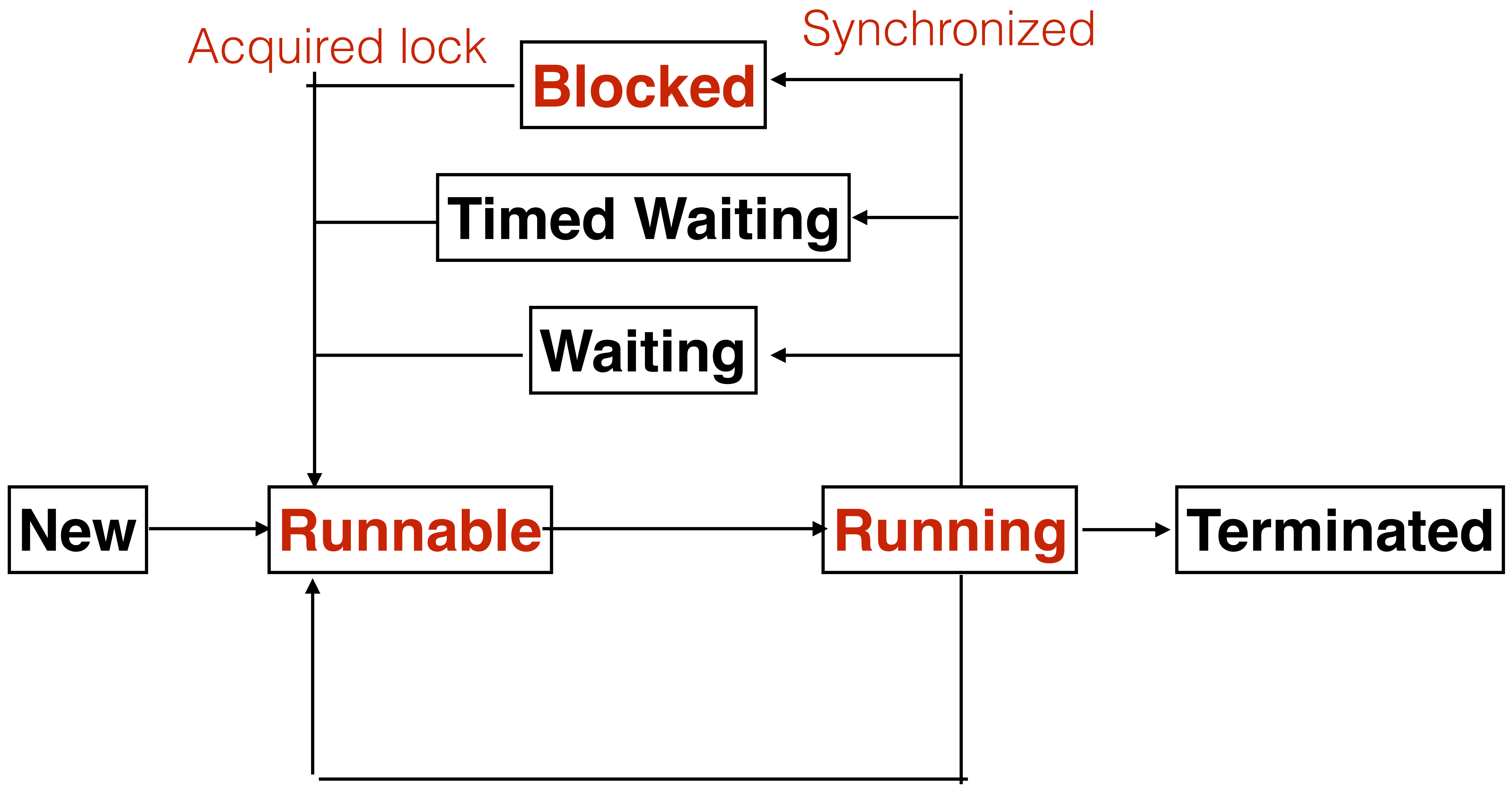
Thread State



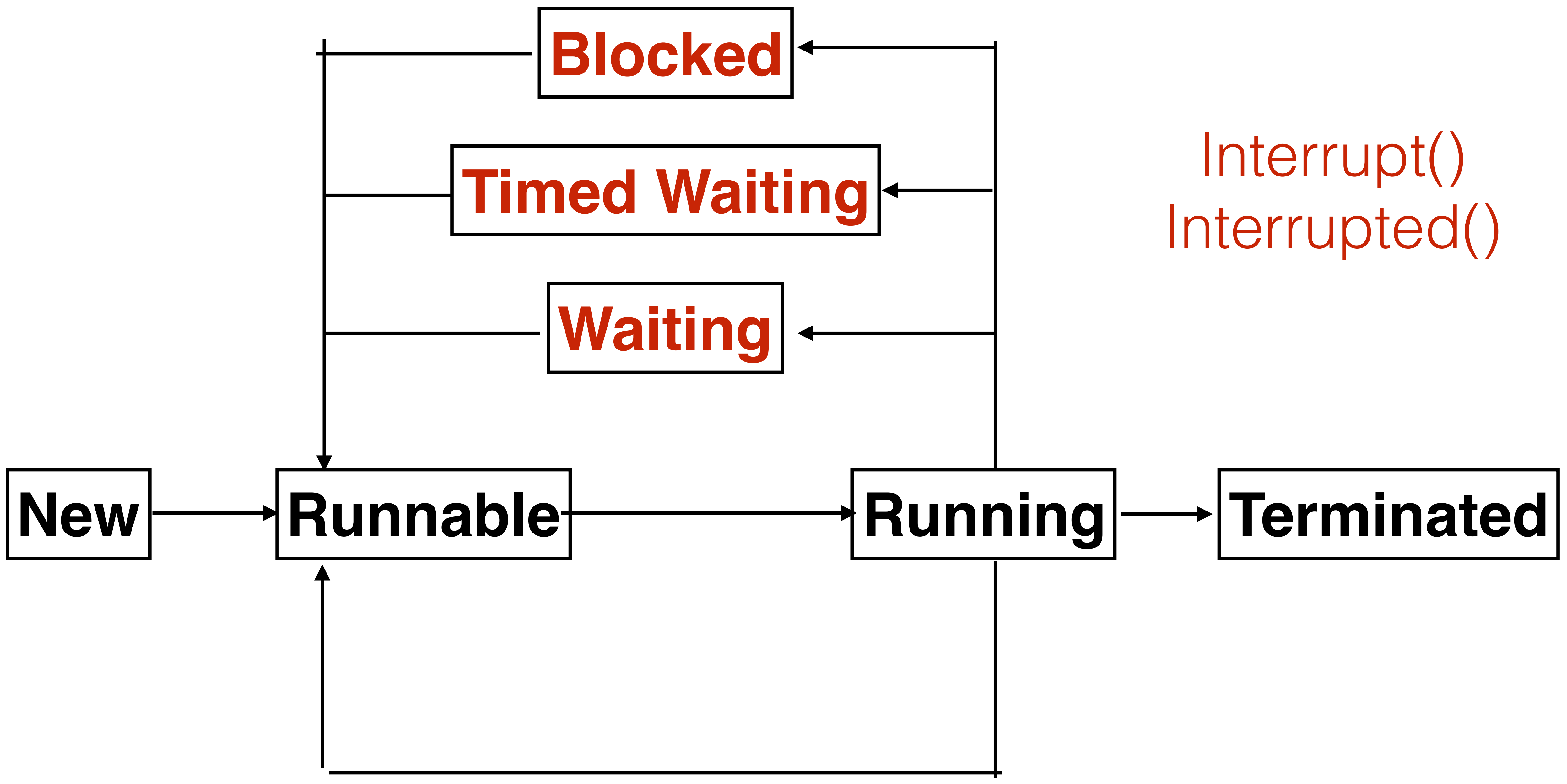
Thread State



Thread State



Thread State



Interrupt()
Interrupted()

Thread Synchronize

- When we should use synchronized
- When we should use wait()
- when multiple threads will modify the field concurrently
- in case of deadlock

Thread Synchronize

```
class station extends Thread{

    static int tickets = 200;

    synchronized void sell() { {tickets - -;} }

    public void run() {while(tickets>0) {sell();}}


    public static void main(String args[]){

        Station stationArray[] = new Station[10];

        for(int i = 0; i < 10; i++)    { stationArray[i] = new Station(); }

  }
```

GUI

- What’s the different Swing and AWT

	AWT	Swing
Devoloper	Sun JDK	Sun JDK
Implemen-tation	Heavy-weighted ; GCD ; Invoke OS Component	Light-weighted ; Top-level container invoke OS component; most component is in pure java
Portablity	Appearance and Behavior depend on OS	Independent with OS
Speed	Fast	Slow before Jdk1.4, but faster now
Component	No abundant	Abundant
Visual Development	No	Jbuilder，Netbeans，Eclipse VE

GUI

- What's the important elements of GUI Programming

- Components
    - different components, Text, Label, Icon, Button..
- Layout
    - how to combine the component
- Action/Event
    - user interaction

GUI

- Containers include
- Top Container JFrame,JDialog
    - connected with OS
    - are not contained in anything else
- intermediate Container JPanel
    - manage component
    - can be nested

GUI

- Components includes

- JTextField
- JButton
- JCheckBox
- JRadioButton
- ….

GUI

- Layouts include

- BorderLayout
- FlowLayout
- GridLayout

GUI

- BorderLayout is the default layout of ?
- BorderLayout rules

- JFrame
- EAST,WEST,NORTH,SOUTH,CENTER

GUI

- FlowLayout is the default layout of ?
- FlowLayout rules

- JPanel
- Left to Right,Up to Down

GUI

- GridLayout Rules

- Row * Column
- Left to Right,Up to Down

GUI

- Event Programming three elements are

- Source : addXXXListener
- handler : xxxeventListener
- event

GUI

- what’s Inner Class priority?
- An inner class gets a special pass to use the outer class' stuff. Even the private stuff.
- they have most of the benefits of a normal class. but with special access rights.

GUI

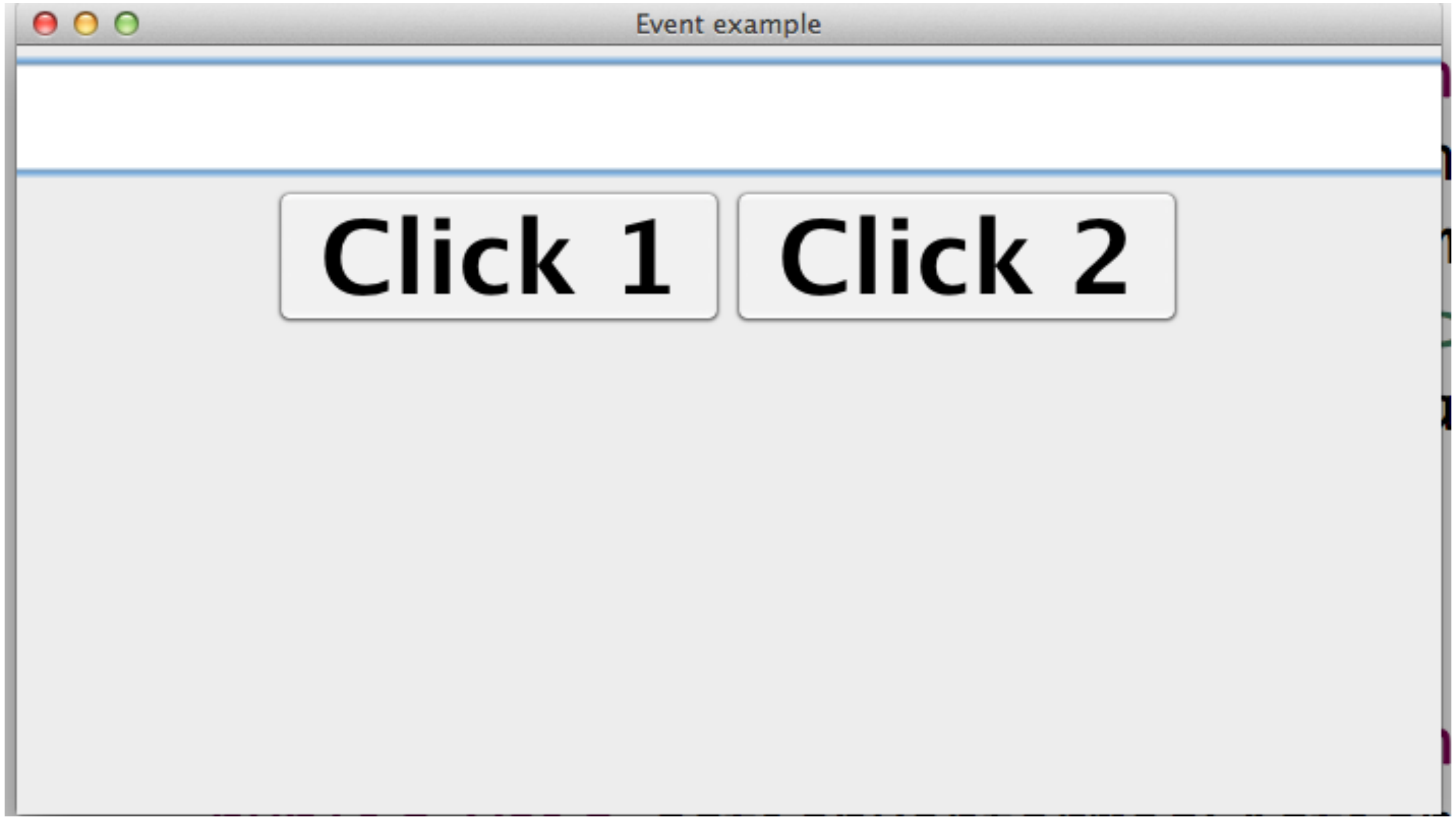
- What's the restriction of inner class

- An Inner class Must be tied to at 1 outer class instance

- you can't new an inner class directly outside the outer cla

GUI

- make a simple action event program
  - ActionListener, addActionListener, JFrame, JButton, JTextFiled



SQL

- CRUD Operation includes
    - Create, Read, Update, Delete
        - Create : create the table and insert data into table
        - Read : select data from table
        - Update : update data in the table
        - Delete : delete data from the table

SQL

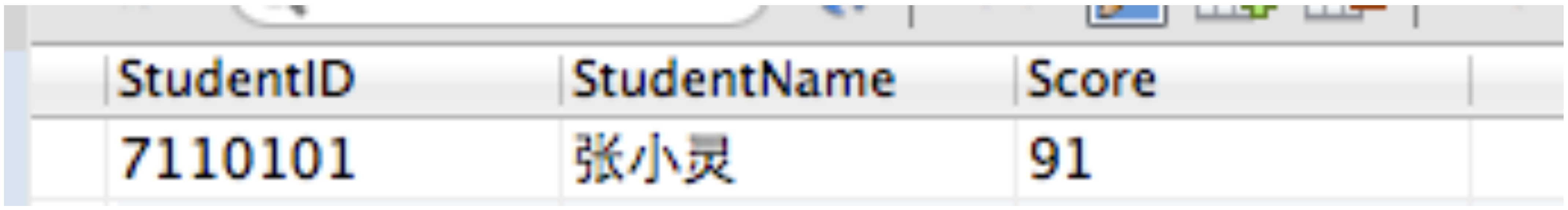
- Data

ColumnName	DataType
StudentName	varchar(10)
StudentID	char(7)
Score	int

- Query

INSERT into Course (StudentName, StudentID, Score) VALUES ("张小灵","7110101", 91) ;

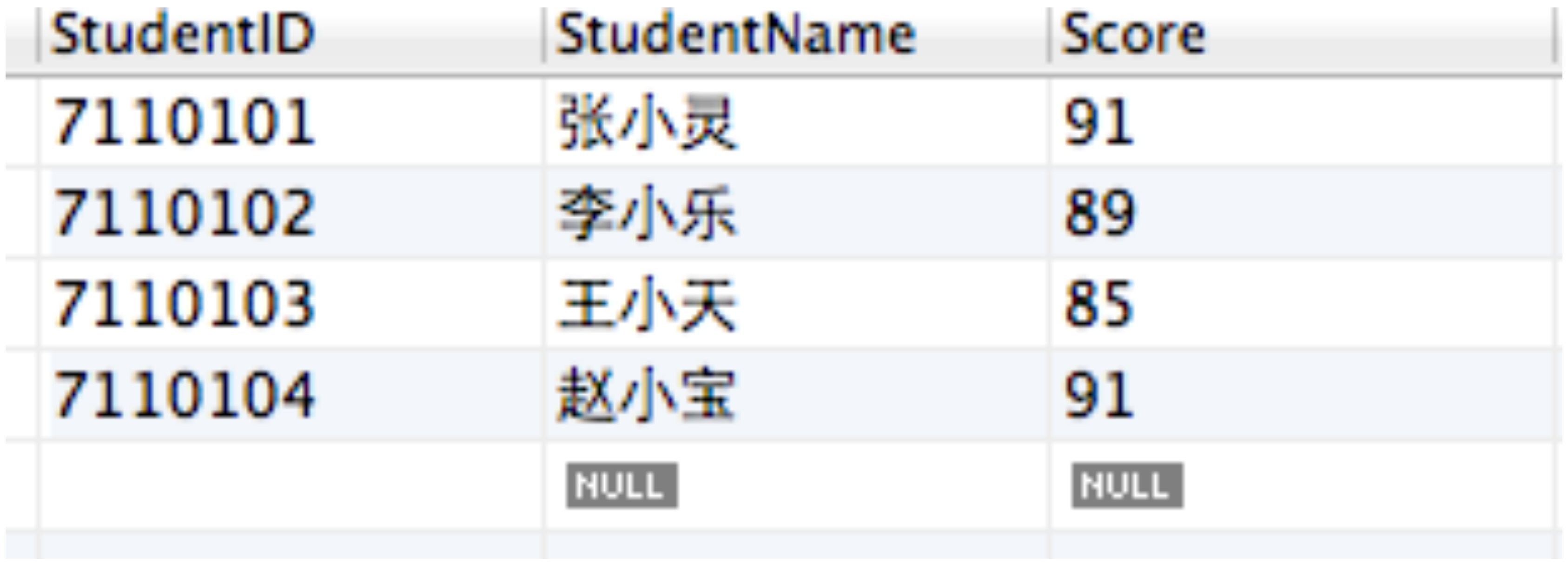
- result



StudentID	StudentName	Score
7110101	张小灵	91

SQL

- Data

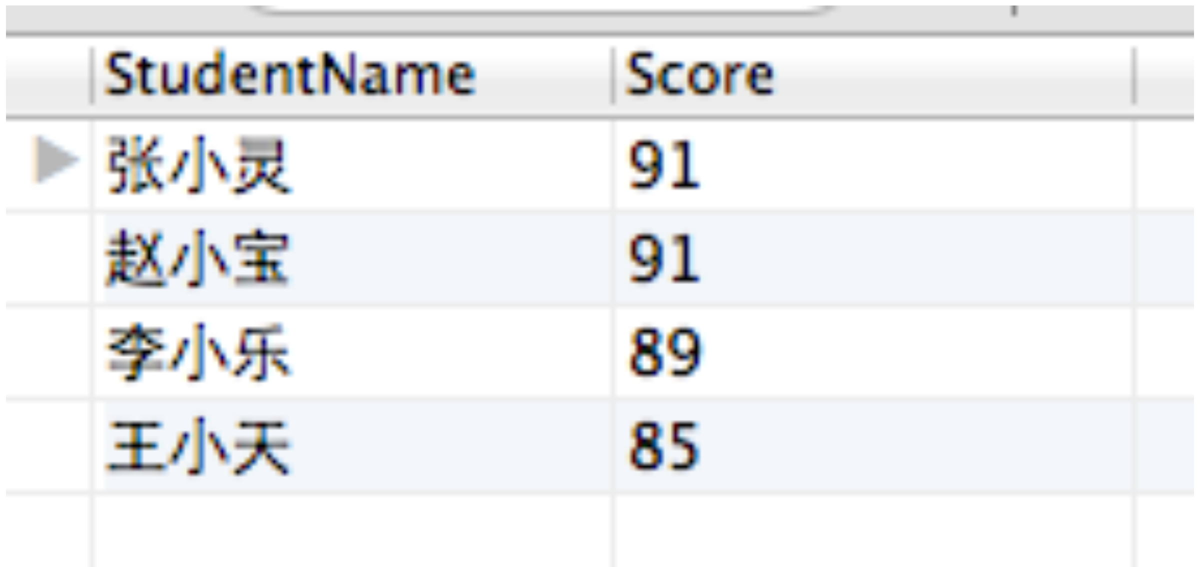


StudentID	StudentName	Score
7110101	张小灵	91
7110102	李小乐	89
7110103	王小天	85
7110104	赵小宝	91
	NULL	NULL

- Query

SELECT StudentName, Score FROM Course Order by Score DESC;

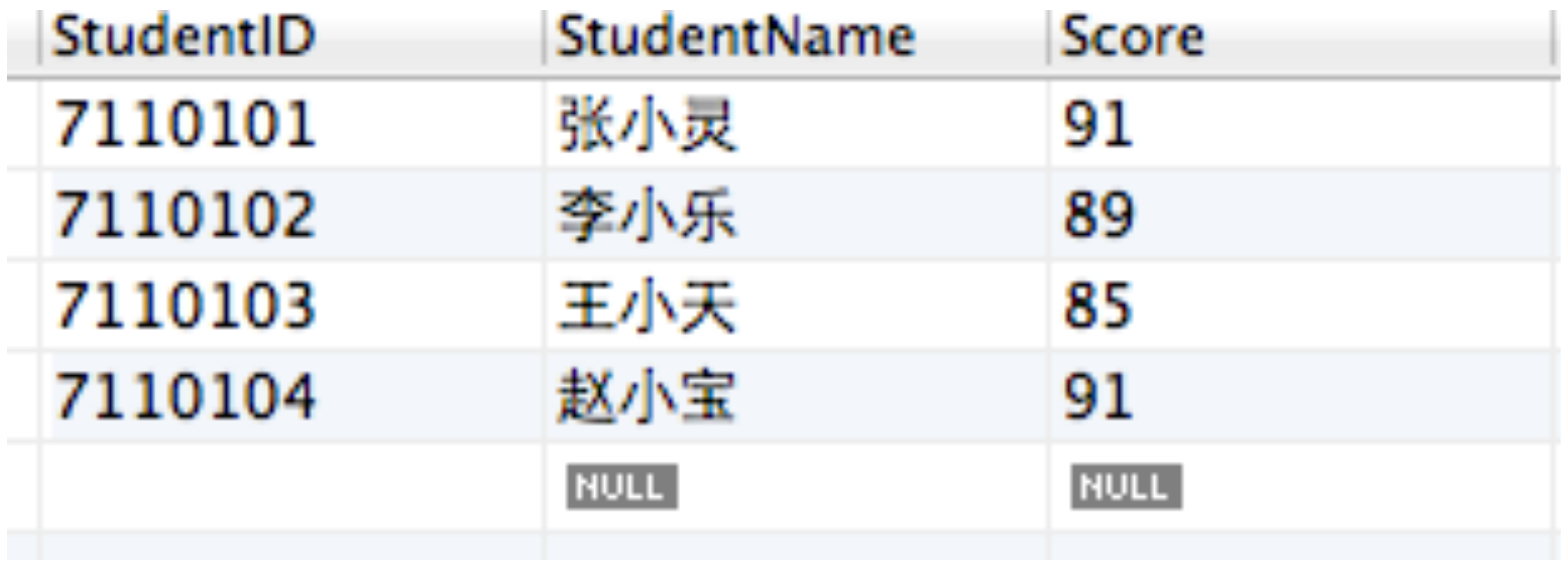
- Result



StudentName	Score
张小灵	91
赵小宝	91
李小乐	89
王小天	85

SQL

- Data

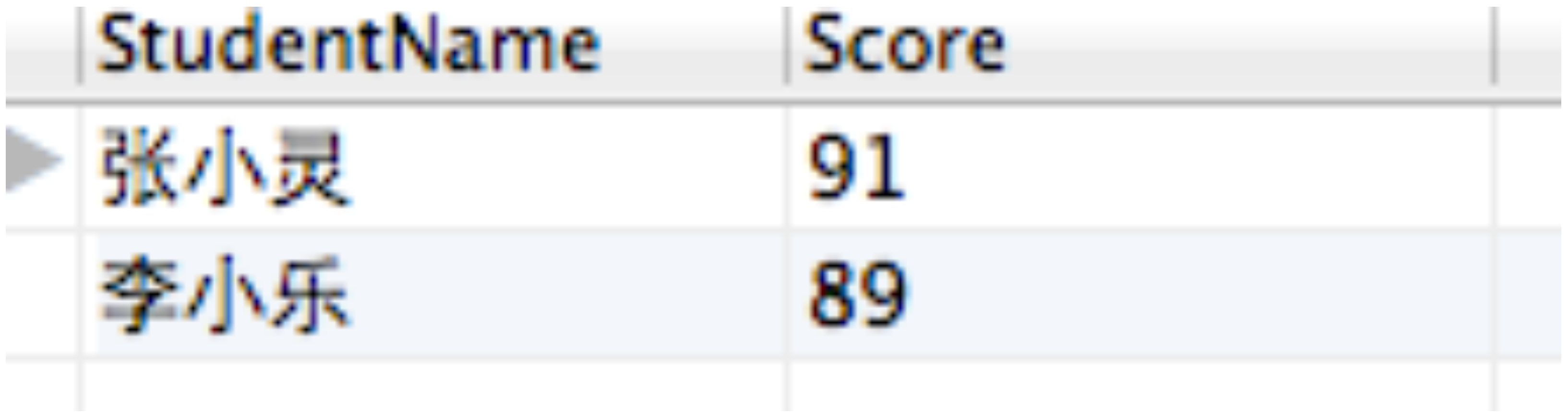


StudentID	StudentName	Score
7110101	张小灵	91
7110102	李小乐	89
7110103	王小天	85
7110104	赵小宝	91
	NULL	NULL

- Query

```
SELECT StudentName, Score FROM Course
Where StudentName in (“张小灵”, “李小乐”);
```

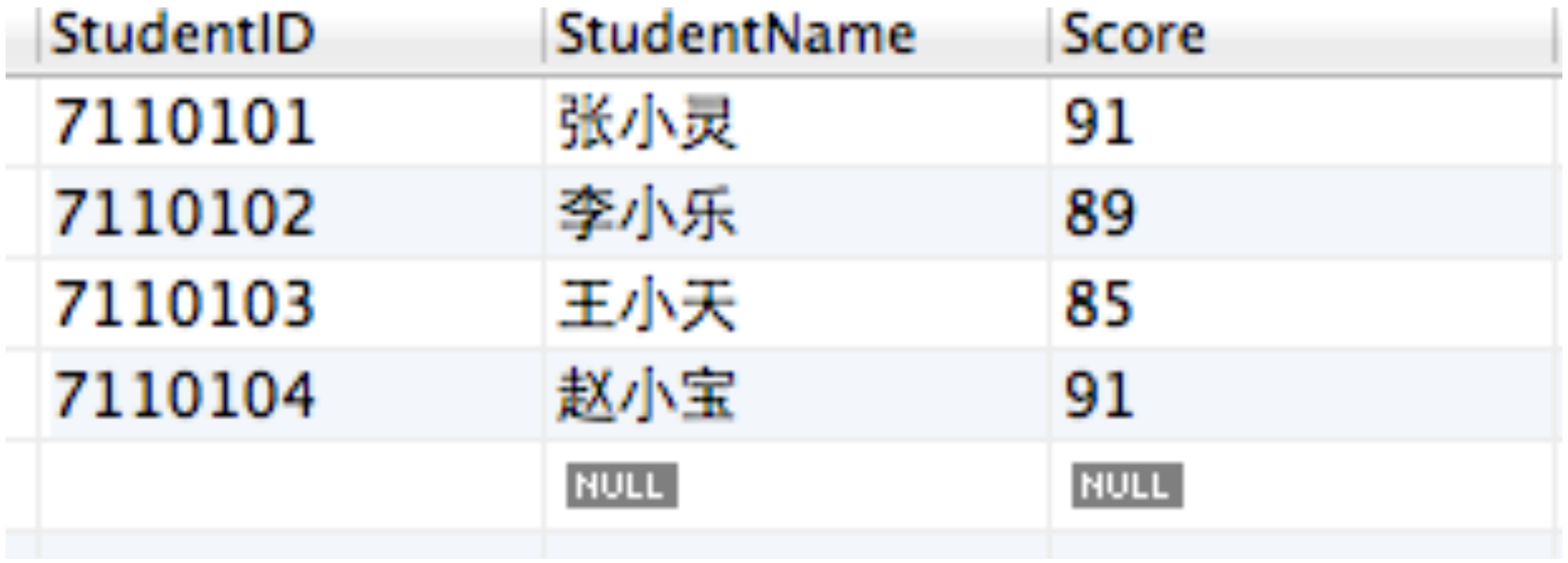
- Result



StudentName	Score
张小灵	91
李小乐	89

SQL

- Data

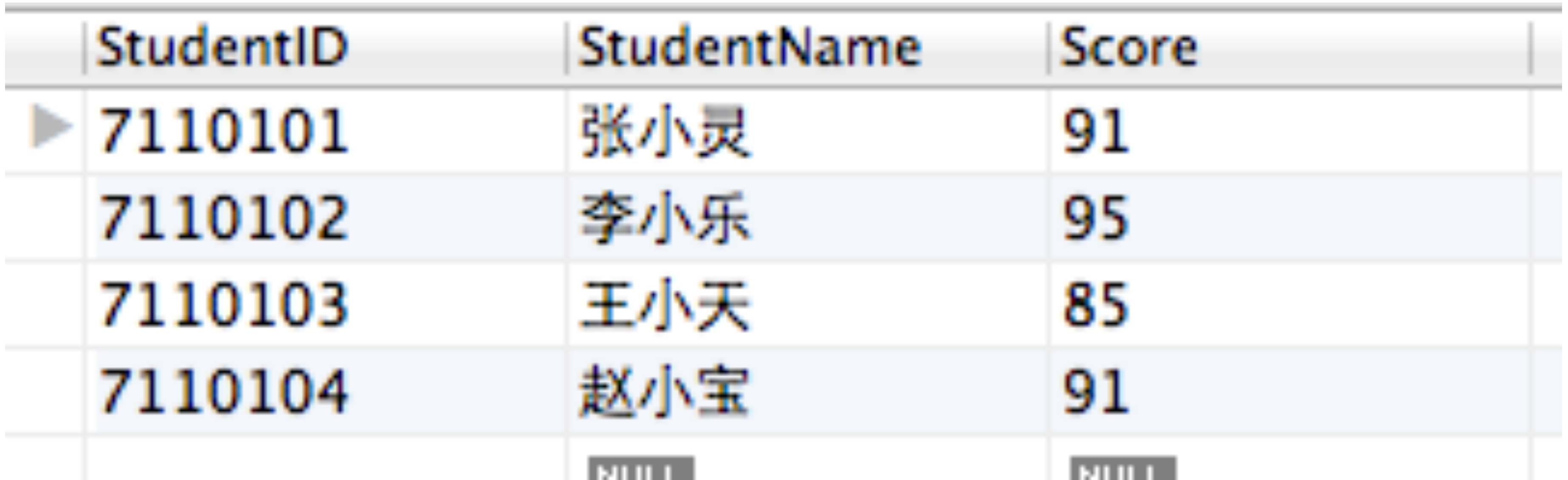


StudentID	StudentName	Score
7110101	张小灵	91
7110102	李小乐	89
7110103	王小天	85
7110104	赵小宝	91
	NULL	NULL

- Query

UPDATE Course SET Score=95 Where StudentID="7110102";

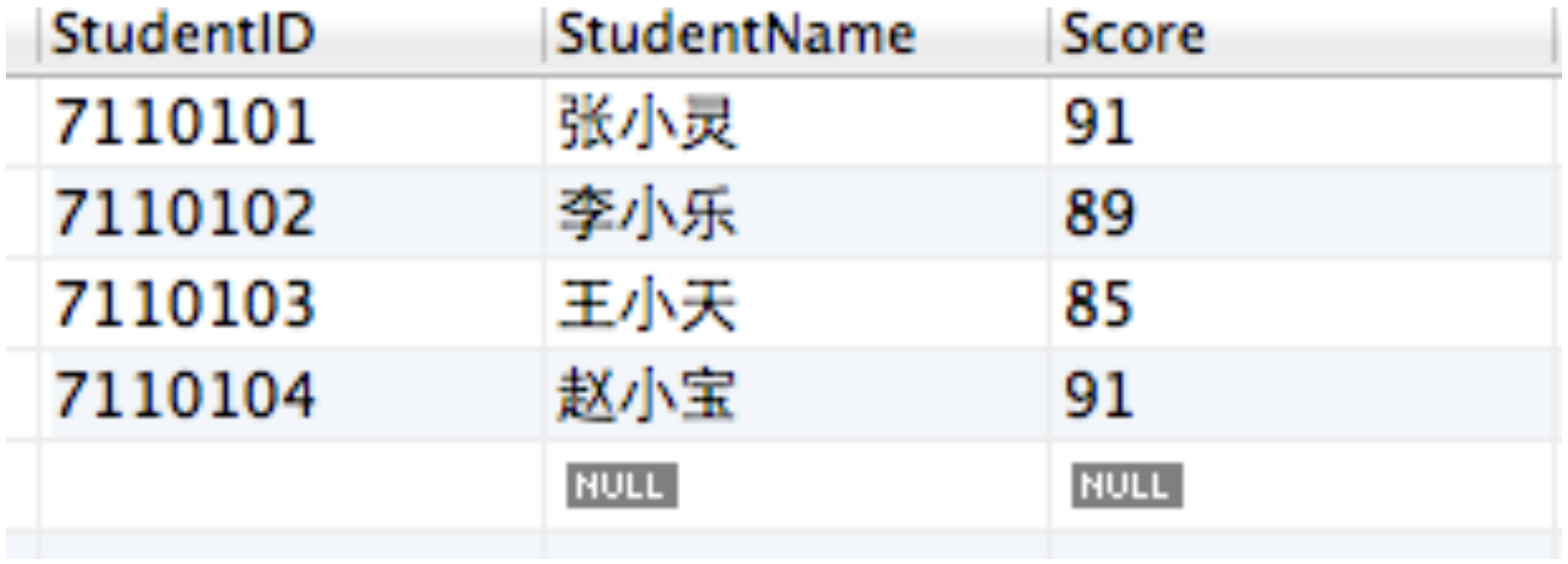
- Result



StudentID	StudentName	Score
7110101	张小灵	91
7110102	李小乐	95
7110103	王小天	85
7110104	赵小宝	91
	NULL	NULL

SQL

- Data

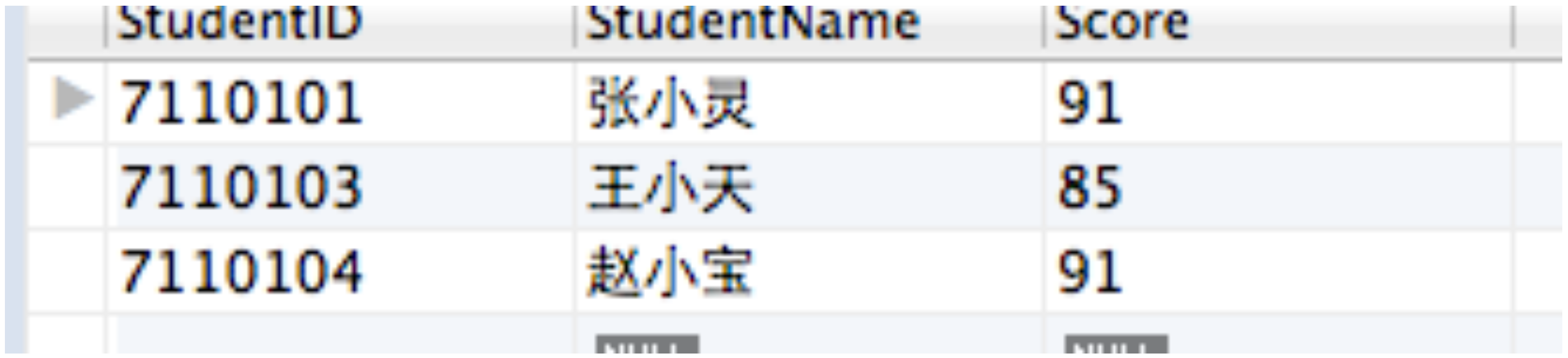


StudentID	StudentName	Score
7110101	张小灵	91
7110102	李小乐	89
7110103	王小天	85
7110104	赵小宝	91
	NULL	NULL

- Query

DELETE FROM Course Where Score>91;

- Result



StudentID	StudentName	Score
7110101	张小灵	91
7110103	王小天	85
7110104	赵小宝	91

SQL

- **import java.sql.*;**
- Load the vendor specific driver
  - [redacted]
    - Dynamically loads a driver class, for Mysql database
- Make the connection
  - **String dbURL =** [redacted]
  - [redacted]
  - **Connection connection = DriverManager.getConnection(dbURL);**
    - Establishes connection to database by obtaining a Connection object

SQL

- Statement have two execute methods, they are:
    - ExecuteQuery
    - ExecuteUpdate

Network

- How to identify a host in network
  - HostName
  - IPAddr
  - DomainName

Network

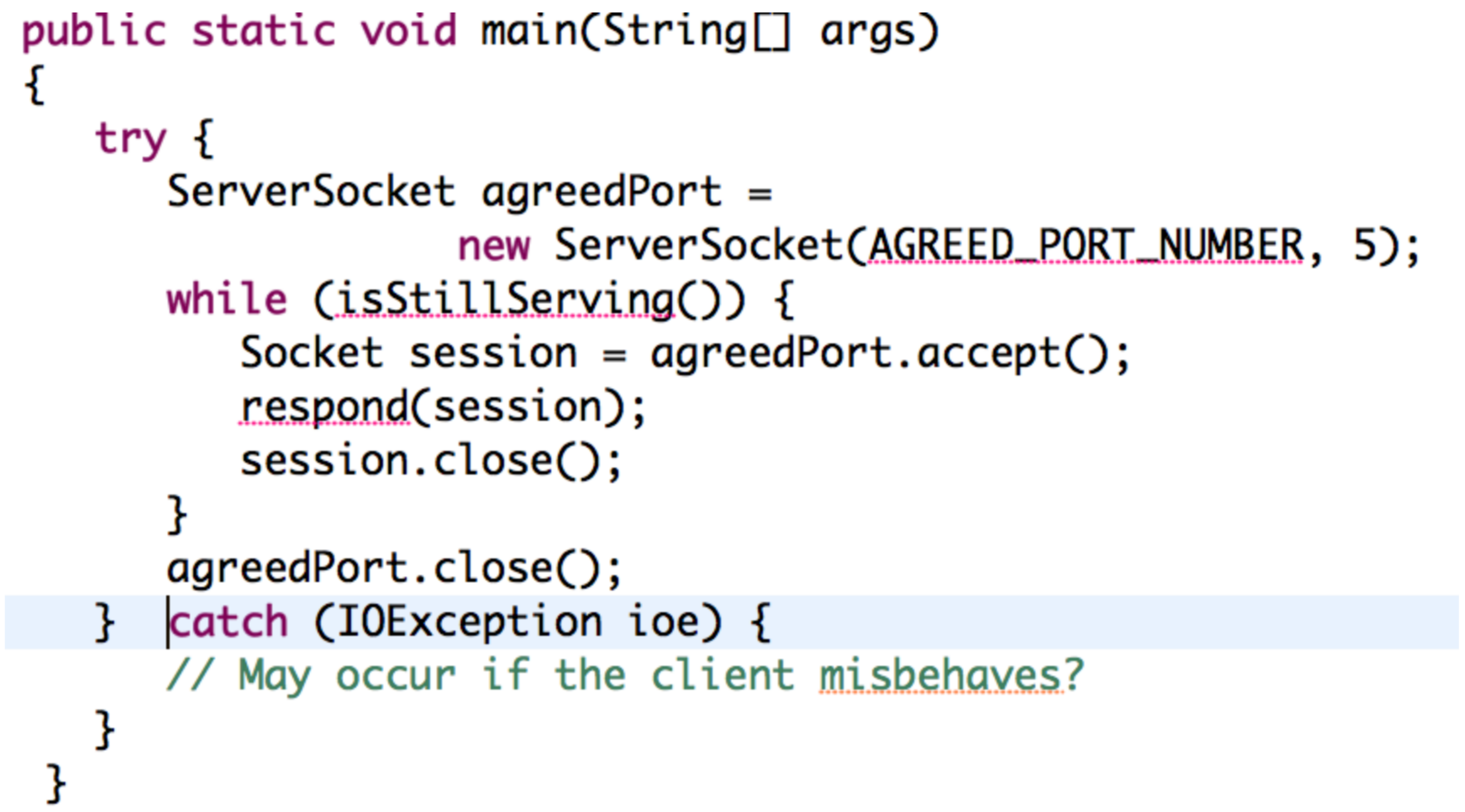
- How to get an INetAddress Object
    - INetAddress.getByName(“www.seu.edu.cn”)
    - INetAddress.getByAnem(“1.2.3.4”)

Network

- There are two types of sockets, they are:
  - ServerSocket
  - Socket

Network

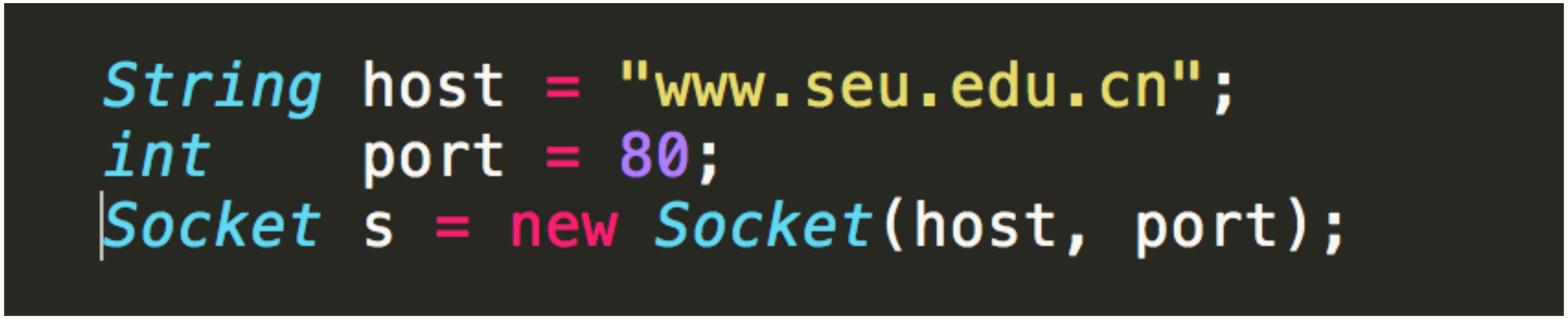
- Setup a ServerSocket at port 80



```
public static void main(String[] args)
{
   try {
      ServerSocket agreedPort =
                  new ServerSocket(AGREED_PORT_NUMBER, 5);
      while (isStillServing()) {
         Socket session = agreedPort.accept();
         respond(session);
         session.close();
      }
      agreedPort.close();
   }  catch (IOException ioe) {
      // May occur if the client misbehaves?
   }
 }
```

Network

- Setup a socket for www.seu.edu.cn port 80



```java
String host = "www.seu.edu.cn";
int    port = 80;
Socket s = new Socket(host, port);
```

Network

- How to read or write to the socket
  - socket.getInputStream()
  - Socket.getOutputStream()