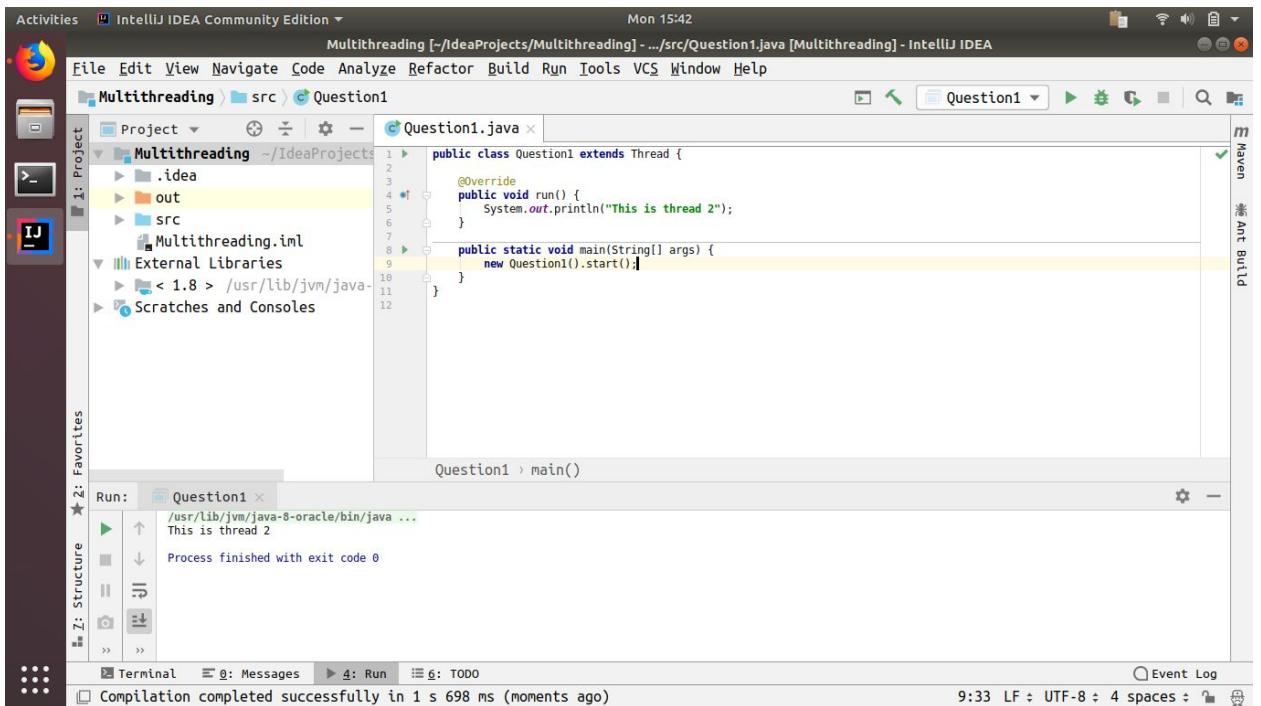


- Create and Run a Thread using Runnable Interface and Thread class.

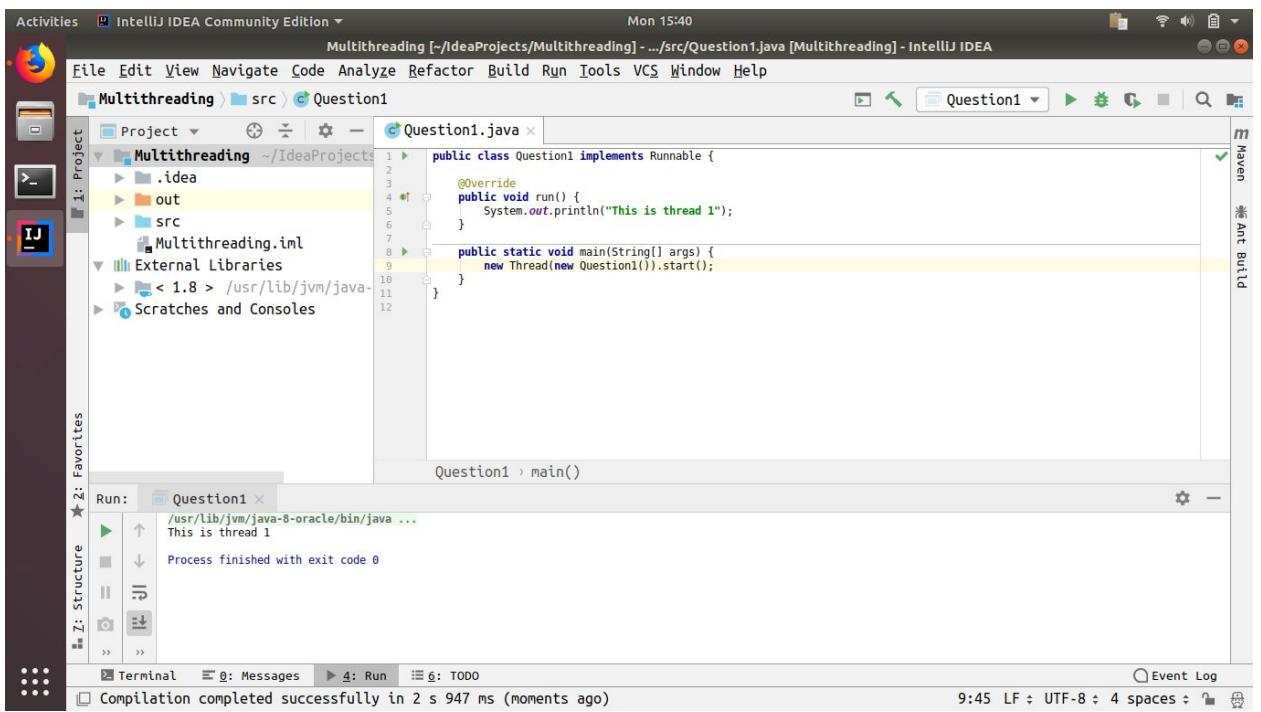


The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "Multithreading". The "src" directory contains a file named "Question1.java".
- Code Editor:** The code implements the `Thread` class:

```
public class Question1 extends Thread {
    @Override
    public void run() {
        System.out.println("This is thread 2");
    }

    public static void main(String[] args) {
        new Question1().start();
    }
}
```
- Run Tab:** The "Run" tab shows the output of running the code. It prints "This is thread 2" and ends with "Process finished with exit code 0".
- Event Log:** The event log shows a successful compilation message: "Compilation completed successfully in 1 s 698 ms (moments ago)".



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "Multithreading". The "src" directory contains a file named "Question1.java".
- Code Editor:** The code implements the `Runnable` interface:

```
public class Question1 implements Runnable {
    @Override
    public void run() {
        System.out.println("This is thread 1");
    }

    public static void main(String[] args) {
        new Thread(new Question1()).start();
    }
}
```
- Run Tab:** The "Run" tab shows the output of running the code. It prints "This is thread 1" and ends with "Process finished with exit code 0".
- Event Log:** The event log shows a successful compilation message: "Compilation completed successfully in 2 s 947 ms (moments ago)".

- Use sleep and join methods with thread.

```
class Thread1 extends Thread {
    @Override
    public void run() {
        try {
            Thread1.sleep( millis: 2000L );
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("This is thread 1");
    }
}

class Thread2 extends Thread {
    @Override
    public void run() {
        try {
            Thread2.sleep( millis: 5000L );
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("This is thread 2");
    }
}

public class Demo{
    public static void main(String[] args) throws InterruptedException {
        Thread t1=new Thread1();
        Thread t2=new Thread2();
        t1.start();
        t1.join();
        t2.start();
        t2.join();
    }
}
```

- Use a `singleThreadExecutor` to submit multiple threads.

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
public class SingleExecutorServiceDemo {
    public static void main(String[] args) {
        ExecutorService executorService = Executors.newSingleThreadExecutor();
        try {
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println("Thread 1");
                }
            });
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println("Thread 2");
                }
            });
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    try {
                        Thread.sleep( millis: 3000L );
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    System.out.println("Thread 3");
                }
            });
        } finally {
            executorService.shutdownNow();
        }
    }
}
```

IntelliJ IDEA Community Edition

Multithreading [-/IdeaProjects/Multithreading] - .../src/SingleExecutorServiceDemo.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SingleExecutorServiceDemo

Project Run: SingleExecutorServiceDemo

/usr/lib/jvm/java-8-oracle/bin/java ...
Thread 1
Thread 2
Thread 3

Process finished with exit code 0

Run TODO Terminal Messages

Compilation completed successfully in 1 s 890 ms (moments ago)

- Try shutdown() and shutdownNow() and observe the difference.

In **shutdown** - is a safe way of shutdown it will wait for all threads to execute after that it will shutdown

IntelliJ IDEA Community Edition

Multithreading [-/IdeaProjects/Multithreading] - .../src/SingleExecutorServiceDemo.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SingleExecutorServiceDemo

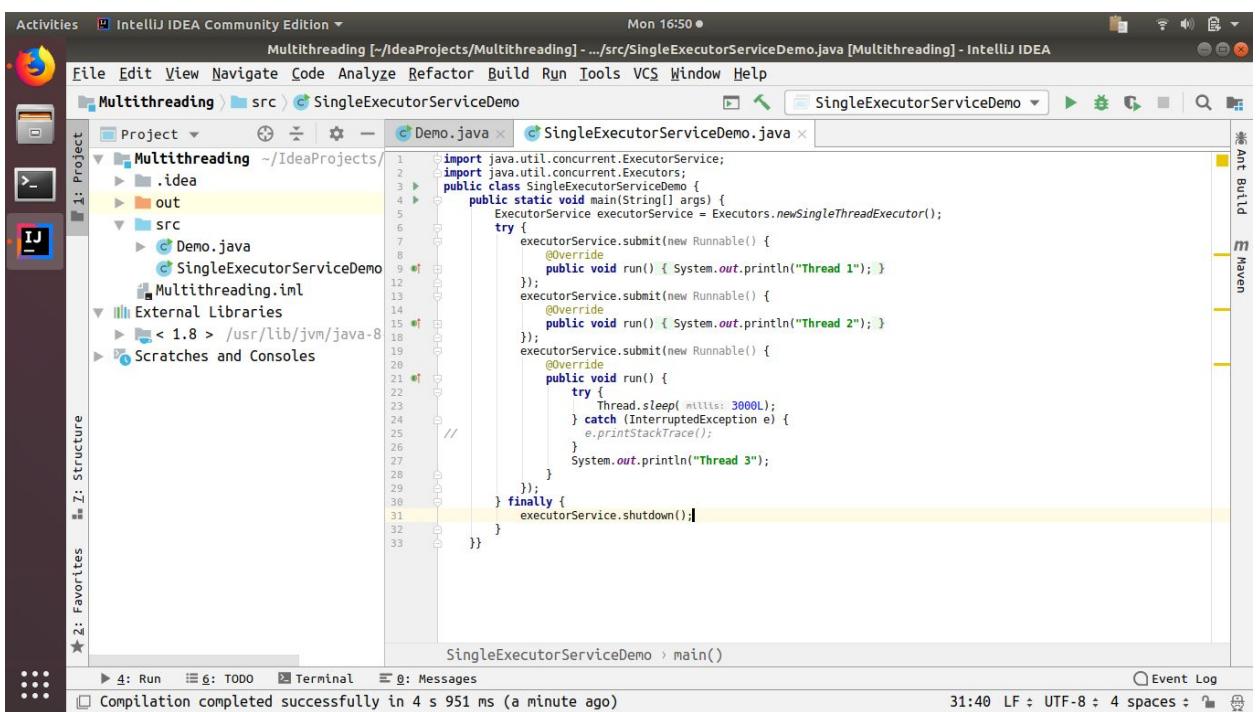
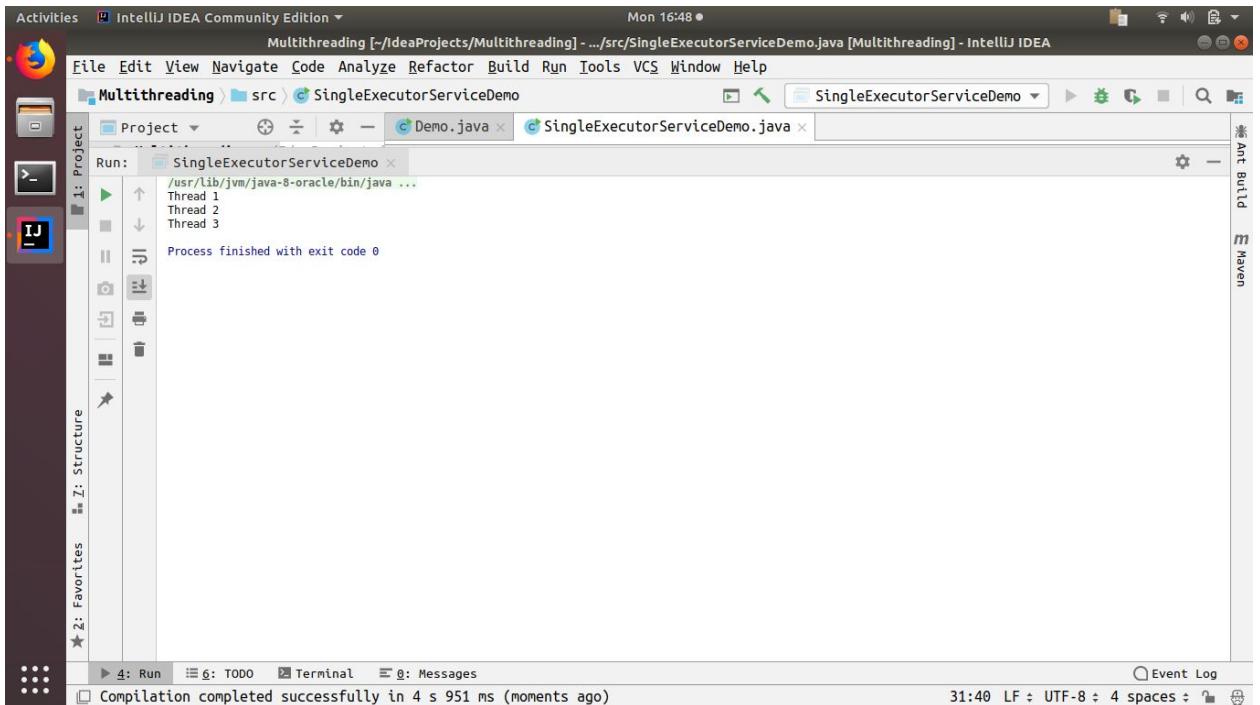
Project Run: SingleExecutorServiceDemo

/usr/lib/jvm/java-8-oracle/bin/java ...
Thread 1
Thread 2

Process finished with exit code 0

Run TODO Terminal Messages

Compilation completed successfully in 4 s 951 ms (moments ago)



Whereas in **shutdownNow()** - it will terminate as soon as it is encountered without waiting for the thread to complete its execution

Activities IntelliJ IDEA Community Edition • Mon 16:34 •

Multithreading [~/IdeaProjects/Multithreading] - .../src/SingleExecutorServiceDemo.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SingleExecutorServiceDemo

Project Multithreading ~/IdeaProjects/ .idea out src

1: Project 2: Favorites 3: Z: Structure

```
1 import java.util.concurrent.ExecutorService;
2 import java.util.concurrent.Executors;
3 public class SingleExecutorServiceDemo {
4     public static void main(String[] args) {
5         ExecutorService executorService = Executors.newSingleThreadExecutor();
6         try {
7             executorService.submit(new Runnable() {
8                 @Override
9                 public void run() {
10                     System.out.println("Thread 1");
11                 }
12             });
13             executorService.submit(new Runnable() {
14                 @Override
15                 public void run() {
16                     System.out.println("Thread 2");
17                 }
18             });
19             executorService.submit(new Runnable() {
20                 @Override
21                 public void run() {
22                     try {
23                         Thread.sleep(3000L);
24                     } catch (InterruptedException e) {
25                         e.printStackTrace();
26                     }
27                     System.out.println("Thread 3");
28                 }
29             });
30         } finally {
31             executorService.shutdownNow();
32         }
33     }
34 }
```

SingleExecutorServiceDemo > main()

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 890 ms (moments ago)

7:1 LF ⚡ UTF-8 ⚡ 4 spaces ⚡ Event Log

Activities IntelliJ IDEA Community Edition • Mon 16:34 •

Multithreading [~/IdeaProjects/Multithreading] - .../src/SingleExecutorServiceDemo.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SingleExecutorServiceDemo

Project Run: SingleExecutorServiceDemo

1: Project 2: Favorites 3: Z: Structure

```
/usr/lib/jvm/java-8-oracle/bin/java ...
Thread 1
Thread 2
Thread 3
```

Process finished with exit code 0

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 890 ms (moments ago)

7:1 LF ⚡ UTF-8 ⚡ 4 spaces ⚡ Event Log

- Use `isShutdown()` and `isTerminated()` with `ExecutorService`.

IntelliJ IDEA Community Edition • Tue 04:17 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/Question5.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Question5

Project 1: Project

```

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Question5 {
    public static void main(String[] args) {
        ExecutorService executorService= Executors.newSingleThreadExecutor();
        try{
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println("Inside thread 1");
                }
            });
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println("Inside thread 2");
                }
            });
            executorService.submit(new Runnable() {
                @Override
                public void run() {
                    System.out.println("Inside thread 3");
                    try {
                        Thread.sleep(5000L);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            });
        }finally {
            executorService.shutdownNow();
            System.out.println("Result of is shutdown"+executorService.isShutdown());
            System.out.println("Result of is terminated "+executorService.isTerminated());
        }
    }
}

```

Question5 > main()

4: Run 6: TODO Terminal 8: Messages

Compilation completed successfully in 1 s 893 ms (moments ago)

Event Log

IntelliJ IDEA Community Edition • Tue 04:17 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/Question5.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Question5

Project 1: Project

Run: Question5

```

/usr/lib/jvm/java-8-oracle/bin/java ...
Inside thread 1
Inside thread 2
Result of is shutdowntrue
Result of is terminated true
Process finished with exit code 0

```

2: Favorites 3: Structure

4: Run 6: TODO Terminal 8: Messages

Compilation completed successfully in 1 s 893 ms (moments ago)

Event Log

- Return a Future from ExecutorService by using callable and use get(), isDone(), isCancelled() with the Future object to know the status of task submitted.

Activities IntelliJ IDEA Community Edition • Tue 04:25 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/Question5.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Question5

Project 1: Project Multithreading ~/IdeaProjects/ .idea out src Demo.java Question5 SingleExecutorServiceDemo.java Multithreading.iml External Libraries < 1.8 > /usr/lib/jvm/java-8- Scratches and Consoles

2: Favorites 2: Structure

Demo.java (selected) SingleExecutorServiceDemo.java Question5.java

```
1 import java.util.concurrent.*;
2 public class Question5 {
3     public static void main(String[] args) throws ExecutionException, InterruptedException {
4         ExecutorService executorService= Executors.newSingleThreadExecutor();
5         Future<Integer> future=executorService.submit(new Callable<Integer>() {
6             @Override
7                 public Integer call() throws Exception {
8                     return 1;
9                 }
10            });
11            try{
12                executorService.submit(new Runnable() {
13                    @Override
14                        public void run() {
15                            System.out.println("Inside thread 1");
16                        }
17                });
18                executorService.submit(new Runnable() {
19                    @Override
20                        public void run() {
21                            System.out.println("Inside thread 2");
22                        }
23                });
24                executorService.submit(new Runnable() {
25                    @Override
26                        public void run() {
27                            System.out.println("Inside thread 3");
28                        }
29                });
30            finally{
31                executorService.shutdown();
32                if(future.isDone()){
33                    System.out.println("Thread have been executed ");
34                    System.out.println(future.get());
35                }
36                if(future.isCancelled()){
37                    System.out.println("threads have been cancelled");
38                }
39            }
40        }
41    }
```

4: Run 6: TODO Terminal 8: Messages

Compilation completed successfully in 1 s 888 ms (a minute ago)

Event Log

Activities IntelliJ IDEA Community Edition • Tue 04:25 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/Question5.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Question5

Project 1: Project Question5

Run: /usr/lib/jvm/java-8-oracle/bin/java ...

Inside thread 1
Inside thread 2
Thread have been executed
1
Inside thread 3

Process finished with exit code 0

2: Favorites 2: Structure

Demo.java SingleExecutorServiceDemo.java Question5.java

4: Run 6: TODO Terminal 8: Messages

Compilation completed successfully in 1 s 888 ms (a minute ago)

Event Log

- Submit List of tasks to ExecutorService and wait for the completion of all the tasks.

Activities IntelliJ IDEA Community Edition ▾ Tue 04:36 ●

Multithreading [-/IdeaProjects/Multithreading] .../src/InvokeAll.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > InvokeAll

Demo.java SingleExecutorServiceDemo.java Question5.java InvokeAll.java

```
1 import java.util.ArrayList;
2 import java.util.List;
3 import java.util.concurrent.*;
4 public class InvokeAll {
5     public static void main(String[] args) throws InterruptedException {
6         List<Callable<Integer>> taskList = new ArrayList<>();
7         taskList.add(new Callable<Integer>() {
8             @Override
9             public Integer call() throws Exception {
10                 return 1;
11             }
12         });
13         taskList.add(new Callable<Integer>() {
14             @Override
15             public Integer call() throws Exception {
16                 return 2;
17             }
18         });
19         taskList.add(new Callable<Integer>() {
20             @Override
21             public Integer call() throws Exception {
22                 return 3;
23             }
24         });
25         ExecutorService executorService= Executors.newSingleThreadExecutor();
26         List<Future<Integer>> futureList= executorService.invokeAll(taskList);
27         futureList.forEach(e)->{
28             if (e.isDone()){
29                 try {
30                     System.out.println("task completed number is "+e.get());
31                 } catch (InterruptedException e1) {
32                     e1.printStackTrace();
33                 } catch (ExecutionException e1) {
34                     e1.printStackTrace();
35                 }
36             }
37         };
38     }
39 }
```

Project Favorites Z: Structure 2: Favorites

Run: TODO Terminal Messages

Compilation completed successfully in 1 s 894 ms (moments ago)

Event Log

Activities IntelliJ IDEA Community Edition ▾ Tue 04:36 ●

Multithreading [-/IdeaProjects/Multithreading] .../src/InvokeAll.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > InvokeAll

Demo.java SingleExecutorServiceDemo.java Question5.java InvokeAll.java

Run: InvokeAll

```
/usr/lib/jvm/java-8-oracle/bin/java ...
task completed number is 1
task completed number is 2
task completed number is 3
```

Project Favorites Z: Structure 2: Favorites

Run: TODO Terminal Messages

Compilation completed successfully in 1 s 894 ms (moments ago)

Event Log

- Schedule task using `schedule()`, `scheduleAtFixedRate()` and `scheduleAtFixedDelay()`

Activities IntelliJ IDEA Community Edition • Tue 05:11 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/Schedular.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Schedular

Project 1: Project

1: Project

out

src

- ↳ Demo.java
- ↳ InvokeAll.java
- ↳ Question5.java
- ↳ Schedular.java
- ↳ ScheduleTask.java
- ↳ SingleExecutorServiceDemo.java

Multithreading.iml

External Libraries

< 1.8 > /usr/lib/jvm/java-8-

Scratches and Consoles

Schedular > main()

```
1 import java.util.concurrent.ExecutorService;
2 import java.util.concurrent.Executors;
3 import java.util.concurrent.ScheduledExecutorService;
4 import java.util.concurrent.TimeUnit;
5
6 public class Schedular {
7     public static void main(String[] args) {
8         ScheduledExecutorService exe= Executors.newSingleThreadScheduledExecutor();
9         exe.scheduleWithFixedDelay(new Thread()->{
10             System.out.println("Running 1st thread");
11         }, InitialDelay: 3L, delay: 2L, TimeUnit.SECONDS);
12     }
13 }
14
```

Run: Schedular x

- ▶
- ↑
- ↓
- »

/usr/lib/jvm/java-8-oracle/bin/java ...
Running 1st thread
Running 1st thread
Running 1st thread
Running 1st thread
Running 1st thread

Process finished with exit code 130 (interrupted by signal 2: SIGINT)

2: Favorites

2: Structure

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 856 ms (moments ago)

11:18 LF UTF-8 4 spaces Event Log

Activities IntelliJ IDEA Community Edition • Tue 05:10 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/Schedular.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Schedular

Project 1: Project

out

src

- ↳ Demo.java
- ↳ InvokeAll.java
- ↳ Question5.java
- ↳ Schedular.java
- ↳ ScheduleTask.java
- ↳ SingleExecutorServiceDemo.java

Multithreading.iml

External Libraries

< 1.8 > /usr/lib/jvm/java-8-

Scratches and Consoles

Schedular > main()

```
1 import java.util.concurrent.ExecutorService;
2 import java.util.concurrent.Executors;
3 import java.util.concurrent.ScheduledExecutorService;
4 import java.util.concurrent.TimeUnit;
5
6 public class Schedular {
7     public static void main(String[] args) {
8         ScheduledExecutorService exe= Executors.newSingleThreadScheduledExecutor();
9         exe.schedule(new Thread()->{
10             System.out.println("Running 1st thread");
11         }, delay: 3L, TimeUnit.SECONDS);
12     }
13 }
14
```

Run: Schedular x

- ▶
- ↑
- ↓
- »

/usr/lib/jvm/java-8-oracle/bin/java ...
Running 1st thread

Process finished with exit code 130 (interrupted by signal 2: SIGINT)

2: Favorites

2: Structure

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 823 ms (moments ago)

11:34 LF UTF-8 4 spaces Event Log

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class Schedular {
    public static void main(String[] args) {
        ScheduledExecutorService exe= Executors.newSingleThreadScheduledExecutor();
        exe.scheduleAtFixedRate(new Thread(()->{
            System.out.println("Running 1st thread");
        }), 1L, 2L, TimeUnit.SECONDS);
    }
}
```

Run: Schedular x

/usr/lib/jvm/java-8-oracle/bin/java ...
Running 1st thread
Running 1st thread
Running 1st thread
Running 1st thread

Process finished with exit code 130 (interrupted by signal 2: SIGINT)

Compilation completed successfully in 1 s 920 ms (moments ago)

- Increase concurrency with Thread pools using newCachedThreadPool() and newFixedThreadPool().

Newcachedpool -

Activities IntelliJ IDEA Community Edition ▾ Tue 05:22 ●

Multithreading [~/IdeaProjects/Multithreading] - .../src/CachedpoolJava [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Cachedpool.java

lon5.java | InvokeAll.java | ScheduleTAsk.java | Schedular.java | Cachedpool.java | Runnable.java

1: Project

```
1 import java.util.concurrent.ExecutorService;
2 import java.util.concurrent.Executors;
3
4 public class Cachedpool implements Runnable {
5     int id;
6     public Cachedpool(int id){
7         this.id=id;
8     }
9     @Override
10    public void run() {
11        System.out.println("thread info - " +Thread.currentThread().getName());
12    }
13}
14 class Demo1{
15     public static void main(String[] args) {
16         ExecutorService exe= Executors.newCachedPool();
17         for(int i=0;i<30;i++){
18             exe.submit(new Cachedpool(i));
19         }
20         exe.shutdown();
21     }
22 }
23
```

2: Favorites 3: Z: Structure

Demo1 > main()

Run: Demo1

Run Terminal Messages

Compilation completed successfully in 1 s 898 ms (a minute ago)

Event Log

Activities IntelliJ IDEA Community Edition ▾ Tue 05:22 ●

Multithreading [~/IdeaProjects/Multithreading] - .../src/CachedpoolJava [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Cachedpool.java

lon5.java | InvokeAll.java | ScheduleTAsk.java | Schedular.java | Cachedpool.java | Runnable.java

1: Project

Run: Demo1

```
/usr/lib/jvm/java-8-oracle/bin/java ...
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-3
thread info - pool-1-thread-3
thread info - pool-1-thread-4
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-5
thread info - pool-1-thread-3
thread info - pool-1-thread-4
thread info - pool-1-thread-1
thread info - pool-1-thread-5
thread info - pool-1-thread-2
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-5
thread info - pool-1-thread-3
thread info - pool-1-thread-7
Process finished with exit code 0
```

2: Favorites 3: Z: Structure

Run Terminal Messages

Compilation completed successfully in 1 s 898 ms (a minute ago)

Event Log

Fixedcachedpool -

Activities IntelliJ IDEA Community Edition ▾ Tue 05:23 ●

Multithreading [~/IdeaProjects/Multithreading] - .../src/CachedpoolJava [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Cachedpool.java

lon5.java | InvokeAll.java | ScheduleTTask.java | Schedular.java | Cachedpool.java | Runnable.java

1: Project

1: 1: Structure

1: 2: Favorites

1: 3: Z: Structure

1: 4: Run

1: 5: TODO

1: 6: Terminal

1: 7: Messages

1: 8: Run: Demo1

1: 9: Art Build

1: 10: Maven

```
1 import java.util.concurrent.ExecutorService;
2 import java.util.concurrent.Executors;
3
4 public class Cachedpool implements Runnable {
5     int id;
6     public Cachedpool(int id){
7         this.id=id;
8     }
9     @Override
10    public void run() {
11        System.out.println("thread info - " +Thread.currentThread().getName());
12    }
13
14    class Demo1{
15        public static void main(String[] args) {
16            ExecutorService exe= Executors.newFixedThreadPool( 4);
17            for(int i=0;i<30;i++){
18                exe.submit(new Cachedpool(i));
19            }
20            exe.shutdown();
21        }
22    }
23}
```

2: Favorites

2: 1: Structure

2: 2: Run: Demo1

2: 3: Z: Structure

2: 4: Run

2: 5: TODO

2: 6: Terminal

2: 7: Messages

2: 8: Event Log

2: 9: Compilation completed successfully in 1 s 767 ms (moments ago)

2: 10: 16:00 LF UTF-8 4 spaces

Activities IntelliJ IDEA Community Edition ▾ Tue 05:23 ●

Multithreading [~/IdeaProjects/Multithreading] - .../src/CachedpoolJava [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > Cachedpool.java

lon5.java | InvokeAll.java | ScheduleTTask.java | Schedular.java | Cachedpool.java | Runnable.java

1: Project

1: 1: Run: Demo1

1: 2: Structure

1: 3: Favorites

1: 4: Z: Structure

1: 5: Run

1: 6: TODO

1: 7: Terminal

1: 8: Messages

1: 9: Event Log

1: 10: Compilation completed successfully in 1 s 767 ms (moments ago)

1: 11: 34:1 LF UTF-8 4 spaces

1: 12: Art Build

1: 13: Maven

```
/usr/lib/jvm/java-8-oracle/bin/java ...
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-2
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-1
thread info - pool-1-thread-2
thread info - pool-1-thread-2
thread info - pool-1-thread-1
Process finished with exit code 0
```

- Use Synchronize method to enable synchronization between multiple threads trying to access method at same time.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Activities, IntelliJ IDEA Community Edition, Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA, Tue 05:29
- Toolbar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project Tool Window:** Shows the project structure with a file named "SynchronisedMethods.java" selected.
- Code Editor:** Displays the Java code for "SynchronisedMethods.java". The code uses synchronized blocks to ensure thread safety when incrementing a shared counter.

```
public class SynchronisedMethods {
    int count;
    synchronized public void increaseCount() {
        count++;
    }
    public void worker1() {
        for (int i = 1; i <=30; i++) {
            increaseCount();
        }
    }
    public void worker2() {
        for (int i = 1; i <=30; i++) {
            increaseCount();
        }
    }
}
public static void main(String[] args) throws InterruptedException {
    SynchronisedMethods synchronisedMethods=new SynchronisedMethods();
    Thread thread1 = new Thread(new Runnable() {
        @Override
        public void run() {
            synchronisedMethods.worker1();
        }
    });
    Thread thread2 = new Thread(new Runnable() {
        @Override
        public void run() {
            synchronisedMethods.worker2();
        }
    });
    thread1.start();
    thread1.join();
    thread2.start();
    thread2.join();
    System.out.println(synchronisedMethods.count);
}
```

- Toolbars:** Standard, Build, Maven.
- Bottom Status Bar:** Compilation completed successfully in 1 s 746 ms (moments ago), 35:7 LF, UTF-8, 4 spaces.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Activities, IntelliJ IDEA Community Edition, Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA, Tue 05:29
- Toolbar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Run Tool Window:** Shows the run configuration for "SynchronisedMethods".
- Code Editor:** Displays the Java code for "SynchronisedMethods.java".
- Toolbars:** Standard, Build, Maven.
- Bottom Status Bar:** Compilation completed successfully in 1 s 818 ms (moments ago), 5:1 LF, UTF-8, 4 spaces.

The run output window shows the command run and the process finished with exit code 0.

- Use Synchronize block to enable synchronization between multiple threads trying to access method at same time.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Activities, IntelliJ IDEA Community Edition, Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA
- Toolbar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help
- Project Tool Window:** Shows the project structure with packages like Multithreading, .idea, out, and src containing files such as ScheduleTTask.java, Schedular.java, Cachedpool.java, and SynchronisedMethods.java.
- Code Editor:** The SynchronisedMethods.java file is open, showing Java code for multithreading using threads and synchronized blocks. The code defines a class SynchronisedMethods with methods increaseCount() and main(). It creates two threads, thread1 and thread2, each executing worker1() or worker2() respectively, and prints the final count value.
- Status Bar:** Compilation completed successfully in 1 s 910 ms (moments ago)

The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Activities, IntelliJ IDEA Community Edition, Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA
- Toolbar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help
- Run Tool Window:** Shows a run configuration for SynchronisedMethods with the command /usr/lib/jvm/java-8-oracle/bin/java ... 60. The status indicates "Process finished with exit code 0".
- Code Editor:** The SynchronisedMethods.java file is open, showing the same Java code as the previous screenshot.
- Status Bar:** Compilation completed successfully in 1 s 910 ms (moments ago)

- Use Atomic Classes instead of Synchronize method and blocks.

Activities IntelliJ IDEA Community Edition • Tue 05:35 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SynchronisedMethods

Project .idea out src

1: Project Multithreading ~/IdeaProjects/Multithreading

2: Favorites Z: Structure

3: Favorites Z: Structure

4: Run 5: TODO 6: Terminal 7: Messages

8: Compilation completed successfully in 1 s 864 ms (a minute ago) 9: Event Log

10: 30:14 LF UTF-8 4 spaces

```
import java.util.concurrent.atomic.AtomicInteger;

public class SynchronisedMethods {
    AtomicInteger count=new AtomicInteger();
    public void increaseCount() {
        synchronized (this){
            count.getAndIncrement();
        }
    }
    public void worker1() {
        for (int i = 1; i <=30; i++) {
            increaseCount();
        }
    }
    public void worker2() {
        for (int i = 1; i <= 30; i++) {
            increaseCount();
        }
    }
    public static void main(String[] args) throws InterruptedException {
        SynchronisedMethods synchronisedMethods=new SynchronisedMethods();
        Thread thread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronisedMethods.worker1();
            }
        });
        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronisedMethods.worker2();
            }
        });
        thread1.start();
        thread1.join();
        thread2.start();
        thread2.join();
        System.out.println(synchronisedMethods.count);
    }
}
```

Activities IntelliJ IDEA Community Edition • Tue 05:35 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SynchronisedMethods

Project Run: SynchronisedMethods

1: Project SynchronisedMethods

2: Favorites Z: Structure

3: Favorites Z: Structure

4: Run 5: TODO 6: Terminal 7: Messages

8: Compilation completed successfully in 1 s 871 ms (moments ago) 9: Event Log

10: 30:14 LF UTF-8 4 spaces

```
/usr/lib/jvm/java-8-oracle/bin/java ...
60
Process finished with exit code 0
```

- Coordinate 2 threads using `wait()` and `notify()`.

Activities IntelliJ IDEA Community Edition • Tue 05:45 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/WaitandNotify.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > WaitandNotify.java

WaitandNotify

Schedular.java Cachedpool.java SynchronisedMethods.java WaitandNotify.java Runnable.java

Project 1: Multithreading

```
1 class Worker{
2     public void worker1(){
3         synchronized (this) {
4             System.out.println("Worker1 Started");
5             try {
6                 wait();
7             } catch (InterruptedException e) {
8                 e.printStackTrace();
9             }
10            System.out.println("Worker1 Done");
11        }
12    }
13    public void worker2(){
14        synchronized (this) {
15            System.out.println("Worker 2 Started");
16            System.out.println("Worker 2 Done");
17            notify();
18        }
19    }
20    public class WaitandNotify {
21        public static void main(String[] args) {
22            Worker worker=new Worker();
23            new Thread(new Runnable() {
24                @Override
25                public void run() {
26                    System.out.println("inside thread 1");
27                    worker.worker1();
28                }
29            }).start();
30            new Thread(new Runnable() {
31                @Override
32                public void run() {
33                    System.out.println("inside thread 2");
34                    worker.worker2();
35                }
36            }).start(); }}
```

WaitandNotify > main() > new Runnable > run()

Run: 4: Run 6: TODO Terminal 8: Messages

Compilation completed successfully in 1 s 822 ms (moments ago)

Event Log

31:52 LF: UTF-8: 4 spaces:

Activities IntelliJ IDEA Community Edition • Tue 05:45 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/WaitandNotify.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > WaitandNotify.java

WaitandNotify

Schedular.java Cachedpool.java SynchronisedMethods.java WaitandNotify.java Runnable.java

Run: WaitandNotify

```
/usr/lib/jvm/java-8-oracle/bin/java ...
inside thread 1
Worker1 Started
inside thread 2
Worker 2 Started
Worker 2 Done
Worker1 Done
```

Process finished with exit code 0

Run: 4: Run 6: TODO Terminal 8: Messages

Compilation completed successfully in 1 s 822 ms (moments ago)

Event Log

31:52 LF: UTF-8: 4 spaces:

- Coordinate multiple threads using `wait()` and `notifyAll()`

Activities IntelliJ IDEA Community Edition ▾ Tue 06:34 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/WaitandNotify.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > WaitandNotify.java

```
1 class Worker{
2     public void worker1(){
3         synchronized (this){
4             try {
5                 System.out.println("Worker1 Started");
6                 wait();
7             } catch (InterruptedException e) {
8                 e.printStackTrace();
9             } finally {
10            }
11            System.out.println("Worker1 Done");
12        }
13    }
14    public void worker2(){
15        synchronized (this){
16            System.out.println("Worker 2 Started");
17            notify();
18            System.out.println("Worker 2 Done");
19        }
20    }
21    public void worker3(){
22        synchronized (this) {
23            System.out.println("Worker 3 Started");
24            System.out.println("Worker 3 Done");
25            notifyAll();
26        }
27    }
28 }
29 public class WaitandNotify {
30     public static void main(String[] args) {
31         Worker worker=new Worker();
32         new Thread(new Runnable() {
33             @Override
34             public void run() {
35                 System.out.println("inside thread 1");
36                 worker.worker1();
37             }
38         }).start();
39         new Thread(new Runnable() {
40             @Override
41             public void run() {
42                 System.out.println("inside thread 2");
43                 worker.worker2();
44             }
45         }).start();
46         new Thread(new Runnable() {
47             @Override
48             public void run() {
49                 System.out.println("inside thread 3");
50                 worker.worker3();
51             }
52         }).start();
53     }
54 }
```

1: Project 2: Favorites 3: Structure

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 766 ms (moments ago)

Event Log

Activities IntelliJ IDEA Community Edition ▾ Tue 06:34 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/WaitandNotify.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > WaitandNotify.java

```
17 public void worker3(){
18     synchronized (this) {
19         System.out.println("Worker 3 Started");
20         System.out.println("Worker 3 Done");
21         notifyAll();
22     }
23 }
24 public class WaitandNotify {
25     public static void main(String[] args) {
26         Worker worker=new Worker();
27         new Thread(new Runnable() {
28             @Override
29             public void run() {
30                 System.out.println("inside thread 1");
31                 worker.worker1();
32             }
33         }).start();
34         new Thread(new Runnable() {
35             @Override
36             public void run() {
37                 System.out.println("inside thread 2");
38                 worker.worker2();
39             }
40         }).start();
41         new Thread(new Runnable() {
42             @Override
43             public void run() {
44                 System.out.println("inside thread 3");
45                 worker.worker3();
46             }
47         }).start();
48     }
49 }
```

1: Project 2: Favorites 3: Structure

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 766 ms (moments ago)

Event Log

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Bar:** Activities, IntelliJ IDEA Community Edition, Tue 06:34 •
- Project Bar:** Multithreading, src, WaitandNotify.java
- Run Tool Window:** Shows the output of the run command for WaitandNotify.java. The output is:

```
/usr/lib/jvm/java-8-oracle/bin/java ...
inside thread 1
Worker1 Started
inside thread 2
Worker 2 Started
Worker 2 Done
Worker1 Done
inside thread 3
Worker 3 Started
Worker 3 Done

Process finished with exit code 0
```
- Bottom Status Bar:** Compilation completed successfully in 1 s 766 ms (moments ago), 13:1 LF, UTF-8, 4 spaces.

- Use Reentrantlock for coordinating 2 threads with signal(), signalAll() and wait().

SIGNAL AND AWAIT

Activities IntelliJ IDEA Community Edition • Tue 06:17 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/SignalAwait.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SignalAwait

Project Multithreading ~/IdeaProjects/Multithreading .idea out src Cachedpool.java Demo.java InvokeAll Question5 Schedular ScheduleTask SignalAwait SignalandAwait SingleExecutorServiceDemo SynchronisedMethods Multithreading.iml External Libraries < 1.8 > /usr/lib/jvm/java-8-Scratches and Consoles

```
public class SignalAwait {
    Lock lock = new ReentrantLock( fair: true );
    Condition condition = lock.newCondition();
    public void worker1() {
        try {
            lock.lock();
            System.out.println("worker 1 Started");
            condition.await();
            System.out.println("worker 1 Finished");
        } catch (InterruptedException e) {
            e.printStackTrace();
        } finally {
            lock.unlock();
        }
    }
    public void worker2() {
        try {
            lock.lock();
            System.out.println("worker 2 Started");
            System.out.println("worker 2 Finished");
            condition.signal();
        } finally {
            lock.unlock();
        }
    }
    public static void main(String[] args) throws InterruptedException {
        SignalAwait signalAwait = new SignalAwait();
        Thread thread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                signalAwait.worker1();
            }
        });
        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                signalAwait.worker2();
            }
        });
        thread1.start();
        thread2.start();
    }
}
```

SignalAwait > main()

Run: 4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 701 ms (moments ago)

Event Log

Activities IntelliJ IDEA Community Edition • Tue 06:17 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/SignalAwait.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SignalAwait

Project Multithreading SignalAwait

Run: /usr/lib/jvm/java-8-oracle/bin/java ...

```
worker 1 Started
worker 2 Started
worker 2 Finished
worker 1 Finished
```

Process finished with exit code 0

Run: 4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 701 ms (moments ago)

Event Log

SIGNAL ALL AND AWAIT

Activities IntelliJ IDEA Community Edition • Tue 06:21 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/SignalAwait.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SignalAwait

Project Multithreading ~/IdeaProjects/1 .idea out src Cachedpool.java Demo.java InvokeAll Question5 Schedular ScheduleTask SignalandAwait.java SignalAwait SingleExecutorServiceDemo SynchronisedMethods Multithreading.iml External Libraries < 1.8 > /usr/lib/jvm/java-8- Scratches and Consoles

```
1 import java.util.concurrent.locks.Condition;
2 import java.util.concurrent.locks.Lock;
3 import java.util.concurrent.locks.ReentrantLock;
4 public class SignalAwait {
5     Lock lock = new ReentrantLock( false );
6     Condition condition = lock.newCondition();
7     public void worker1() {
8         try {
9             lock.lock();
10            System.out.println("worker 1 Started");
11            condition.await();
12            System.out.println("worker 1 Finished");
13        } catch ( InterruptedException e ) {
14            e.printStackTrace();
15        } finally {
16            lock.unlock();
17        }
18    }
19    public void worker2() {
20        try {
21            lock.lock();
22            System.out.println("worker 2 Started");
23            condition.await();
24            System.out.println("worker 2 Finished");
25        } catch ( InterruptedException e ) {
26            e.printStackTrace();
27        } finally {
28            lock.unlock();
29        }
30    }
31    public void worker3() {
32        try {
33            lock.lock();
34            System.out.println("worker 3 Started");
35            System.out.println("worker 3 Finished");
36            condition.signalAll();
37        } finally {
38            lock.unlock();
39        }
40    }
41    public static void main(String[] args) throws InterruptedException {
42        SignalAwait signalAwait = new SignalAwait();
43        Thread thread1 = new Thread(new Runnable() {
44            @Override
45            public void run() {
46                signalAwait.worker1();
47            }
48        });
49        Thread thread2 = new Thread(new Runnable() {
50            @Override
51            public void run() {
52                signalAwait.worker2();
53            }
54        });
55        Thread thread3 = new Thread(new Runnable() {
56            @Override
57            public void run() {
58                signalAwait.worker3();
59            }
60        });
61        thread1.start();
62        thread2.start();
63        thread3.start();
64    }
65 }
```

SignalAwait > worker2()

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 814 ms (moments ago)

22:41 LF UTF-8 4 spaces Event Log

Activities IntelliJ IDEA Community Edition • Tue 06:21 •

Multithreading [-/IdeaProjects/Multithreading] - .../src/SignalAwait.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SignalAwait

Project Multithreading ~/IdeaProjects/1 .idea out src Cachedpool.java Demo.java InvokeAll Question5 Schedular ScheduleTask SignalandAwait.java SignalAwait SingleExecutorServiceDemo SynchronisedMethods Multithreading.iml External Libraries < 1.8 > /usr/lib/jvm/java-8- Scratches and Consoles

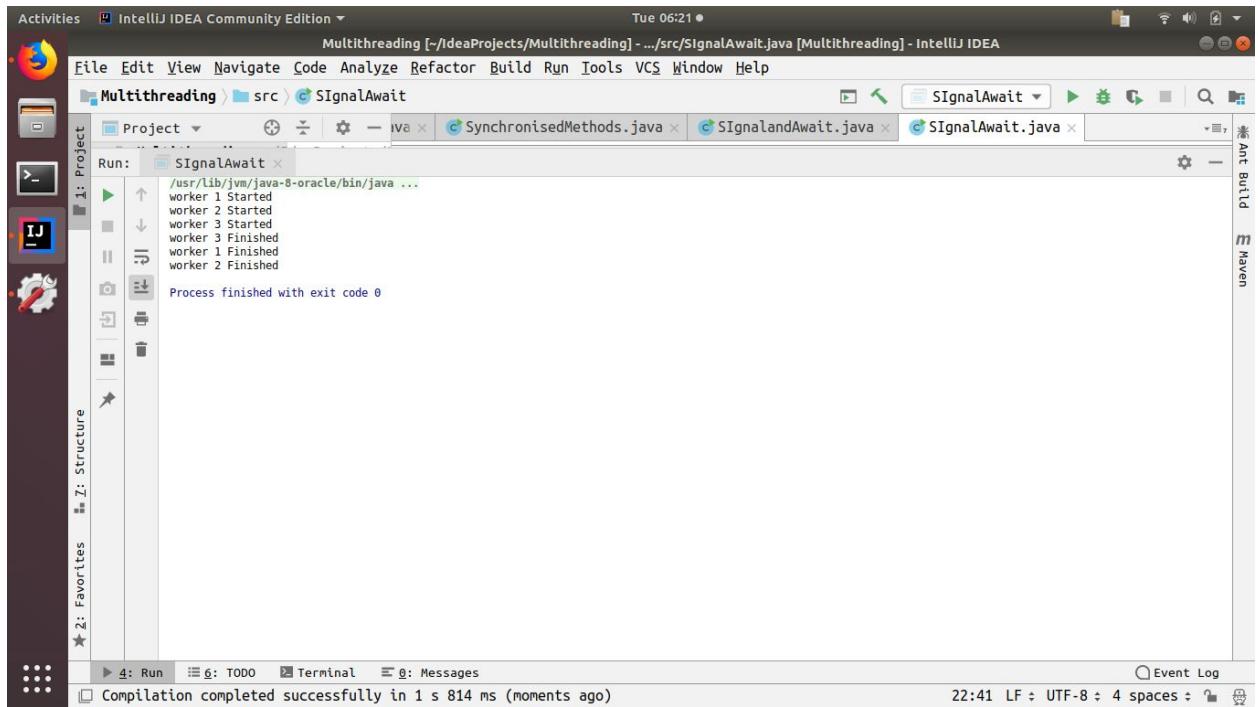
```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54 }
```

SignalAwait > worker2()

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 814 ms (moments ago)

22:41 LF UTF-8 4 spaces Event Log



- Create a deadlock and Resolve it using tryLock().

Activities IntelliJ IDEA Community Edition ▾ Tue 06:47 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SynchronisedMethods

SynchronisedMethods.java

```
1: 1: Project
2: Multithreading.java x Cachedpool.java x SynchronisedMethods.java x WaitandNotify.java x SignalAwait.java x Deadlock.java x
3: 2: Favorites 3: Z: Structure
4: 5: 6: public class SynchronisedMethods {
5:     int count;
6:     ReentrantLock lock1=new ReentrantLock();
7:     ReentrantLock lock2=new ReentrantLock();
8: 
9:     public void worker1() {
10:         lock1.lock();
11:         lock2.lock();
12:         System.out.println("locks taken by thread 1 ");
13:     }
14: 
15:     public void worker2() {
16:         lock2.lock();
17:         lock1.lock();
18:         System.out.println("locks taken by thread 2");
19:     }
20: 
21:     public static void main(String[] args) throws InterruptedException {
22:         SynchronisedMethods synchronisedMethods=new SynchronisedMethods();
23:         new Thread(new Runnable() {
24:             @Override
25:             public void run() {
26:                 synchronisedMethods.worker1();
27:             }
28:         }).start();
29:         new Thread(new Runnable() {
30:             @Override
31:             public void run() {
32:                 synchronisedMethods.worker2();
33:             }
34:         }).start();
35:     }
36: }
```

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 987 ms (moments ago)

3:1 LF ⚡ UTF-8 ⚡ 4 spaces ⚡ Event Log

Activities IntelliJ IDEA Community Edition ▾ Tue 06:47 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SynchronisedMethods

Run: SynchronisedMethods

```
/usr/lib/jvm/java-8-oracle/bin/java ...
locks taken by thread 1
```

1: Project
2: Favorites 3: Z: Structure

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 987 ms (moments ago)

3:1 LF ⚡ UTF-8 ⚡ 4 spaces ⚡ Event Log

USING TRYLOCK -

Activities IntelliJ IDEA Community Edition ▾ Tue 06:50 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SynchronisedMethods

1: Project 2: Favorites 3: Z: Structure

```
sheduler.java x Cachedpool.java x SynchronisedMethods.java x WaitandNotify.java x SignalAwait.java x Deadlock.java x
```

1 import java.util.concurrent.locks.ReentrantLock;
2 public class SynchronisedMethods {
3 int count;
4 ReentrantLock lock1=new ReentrantLock();
5 ReentrantLock lock2=new ReentrantLock();
6 public void acquireLock(ReentrantLock lock1, ReentrantLock lock2){
7 boolean acquireLock1= lock1.tryLock();
8 boolean acquireLock2= lock2.tryLock();
9 if(acquireLock1 && acquireLock2){
10 return;
11 }
12 if(acquireLock1){
13 lock1.unlock();
14 }
15 if(acquireLock2){
16 lock2.unlock();
17 }
18 }
19 public void worker1(){
20 acquireLock(lock1,lock2);
21 System.out.println("lock 1 worker 1");
22 System.out.println("lock 2 worker 1");
23 lock2.unlock();
24 lock1.unlock();
25 }
26 public void worker2(){
27 acquireLock(lock2,lock1);
28 System.out.println("lock 1 worker 2");
29 System.out.println("lock 2 worker 2");
30 lock1.unlock();
31 lock2.unlock();
32 }
33 public static void main(String[] args) throws InterruptedException {
34 SynchronisedMethods

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 957 ms (moments ago)

18:6 LF ⚡ UTF-8 ⚡ 4 spaces ⚡ Event Log

Activities IntelliJ IDEA Community Edition ▾ Tue 06:50 ●

Multithreading [-/IdeaProjects/Multithreading] - .../src/SynchronisedMethods.java [Multithreading] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Multithreading > src > SynchronisedMethods

1: Project 2: Favorites 3: Z: Structure

```
sheduler.java x Cachedpool.java x SynchronisedMethods.java x WaitandNotify.java x SignalAwait.java x Deadlock.java x
```

19 public void worker1(){
20 acquireLock(lock1,lock2);
21 System.out.println("lock 1 worker 1");
22 System.out.println("lock 2 worker 1");
23 lock2.unlock();
24 lock1.unlock();
25 }
26 public void worker2(){
27 acquireLock(lock2,lock1);
28 System.out.println("lock 1 worker 2");
29 System.out.println("lock 2 worker 2");
30 lock1.unlock();
31 lock2.unlock();
32 }
33 public static void main(String[] args) throws InterruptedException {
34 SynchronisedMethods synchronisedMethods=new SynchronisedMethods();
35 new Thread(new Runnable() {
36 @Override
37 public void run() {
38 synchronisedMethods.worker1();
39 }
40 }).start();
41 new Thread(new Runnable() {
42 @Override
43 public void run() {
44 synchronisedMethods.worker2();
45 }
46 }).start();
47 }
48 }

4: Run 5: TODO 6: Terminal 7: Messages

Compilation completed successfully in 1 s 957 ms (moments ago)

18:6 LF ⚡ UTF-8 ⚡ 4 spaces ⚡ Event Log

