Question: 1:

Write a program to demonstrate the use of volatile keyword.

```java
package javaDay3;

import java.util.Scanner;

class Processor extends Thread{
    private volatile boolean running = true;

    public void Run(){
        while(running){
            System.out.println("hello");

            try {
                Thread.sleep( millis: 100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public void shutdown(){
        running = false;
    }
}

public class Question1 {

    public static void main(String[] args) {
        Processor p1 = new Processor();
        p1.start();
```

```java
        }
    }

public class Question1 {

    public static void main(String[] args) {
        Processor p1 = new Processor();
        p1.start();
        System.out.println("Press return to stop--------------");

        Scanner s = new Scanner(System.in);
        s.nextLine();

        p1.shutdown();
    }
}
```

```
/home/parul/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Press return to stop--------------
```
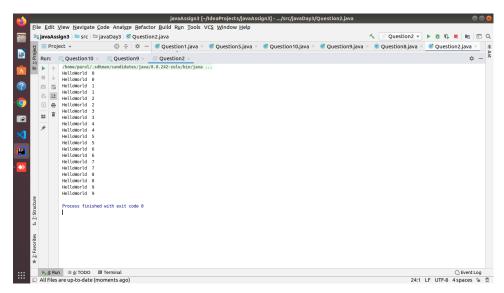
Question: 2:

Write a program to create a thread using Thread class and Runnable interface each.

```java
package javaDay3;

class Runner extends Thread{
    public void run(){
        for (int i = 0; i < 10; i++){
            System.out.println("HelloWorld  " + i);

            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }
    }
}

class Runner2 implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println("HelloWorld  " + i);

            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

    }
```

Question2 > main()

```java
        @Override
        public void run() {
            for (int i = 0; i < 10; i++) {
                System.out.println("HelloWorld  " + i);

                try {
                    Thread.sleep( millis: 1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }

        }
}

public class Question2 {
    public static void main(String[] args) {
        Runner runner1 = new Runner();

        Thread t1 = new Thread(new Runner2());
        runner1.start();
        t1.start();

    }
}
```

Question2 > main()

```
/home/parul/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
HelloWorld  0
HelloWorld  0
HelloWorld  1
HelloWorld  1
HelloWorld  2
HelloWorld  2
HelloWorld  2
HelloWorld  3
HelloWorld  3
HelloWorld  4
HelloWorld  4
HelloWorld  5
HelloWorld  5
HelloWorld  6
HelloWorld  6
HelloWorld  7
HelloWorld  7
HelloWorld  8
HelloWorld  8
HelloWorld  9
HelloWorld  9

Process finished with exit code 0
```
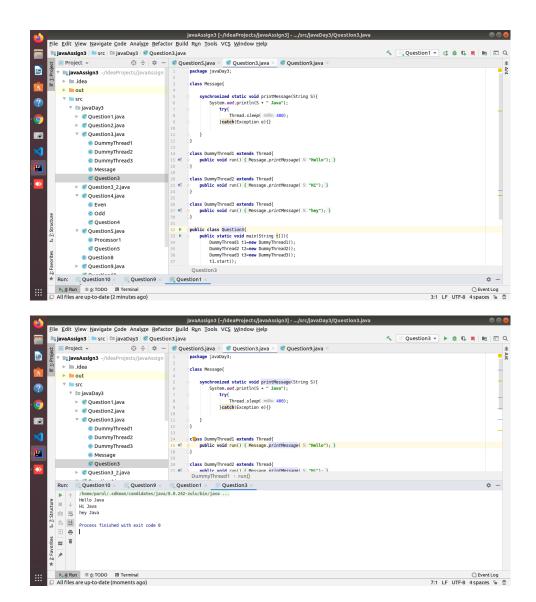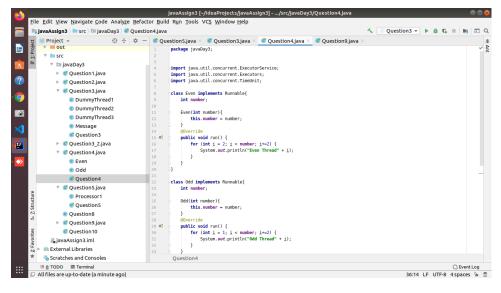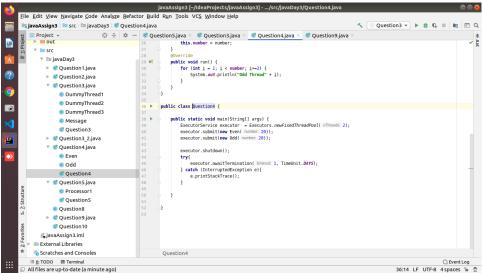
Question: 3:
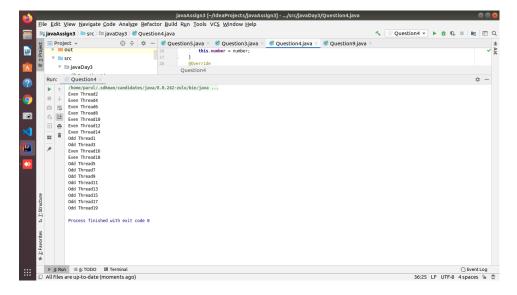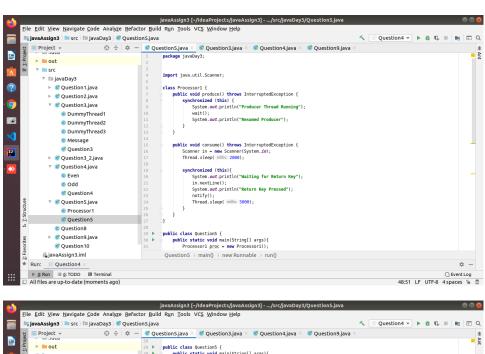Write a program using synchronization block and synchronization method.

Question: 4:
Write a program to create a Thread pool of 2 threads where one Thread will print even numbers and other will print odd numbers.
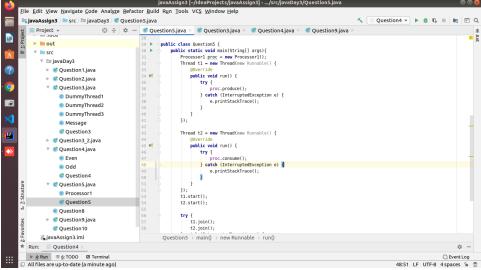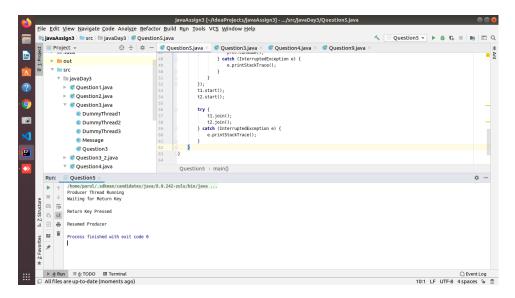
Screenshot 1:

```java
package javaDay3;


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

class Even implements Runnable{
    int number;

    Even(int number){
        this.number = number;
    }
    @Override
    public void run() {
        for (int i = 2; i < number; i+=2) {
            System.out.println("Even Thread" + i);
        }
    }
}

class Odd implements Runnable{
    int number;

    Odd(int number){
        this.number = number;
    }
    @Override
    public void run() {
        for (int i = 1; i < number; i+=2) {
            System.out.println("Odd Thread" + i);
        }
    }
}
```

Screenshot 2:

```java
        this.number = number;
    }
    @Override
    public void run() {
        for (int i = 1; i < number; i+=2) {
            System.out.println("Odd Thread" + i);
        }
    }
}

public class Question4 {

    public static void main(String[] args) {
        ExecutorService executor = Executors.newFixedThreadPool( nThreads: 2);
        executor.submit(new Even( number: 20));
        executor.submit(new Odd( number: 20));

        executor.shutdown();
        try{
            executor.awaitTermination( timeout: 1, TimeUnit.DAYS);
        } catch (InterruptedException e){
            e.printStackTrace();
        }
    }
}
```

Screenshot 3 (Run output):

```
/home/parul/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Even Thread2
Even Thread4
Even Thread6
Even Thread8
Even Thread10
Even Thread12
Even Thread14
Odd Thread1
Odd Thread3
Even Thread16
Even Thread18
Odd Thread5
Odd Thread7
Odd Thread9
Odd Thread11
Odd Thread13
Odd Thread15
Odd Thread17
Odd Thread19

Process finished with exit code 0
```

Question: 5:

Write a program to demonstrate wait and notify methods.
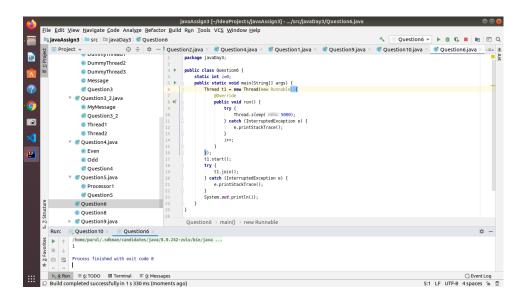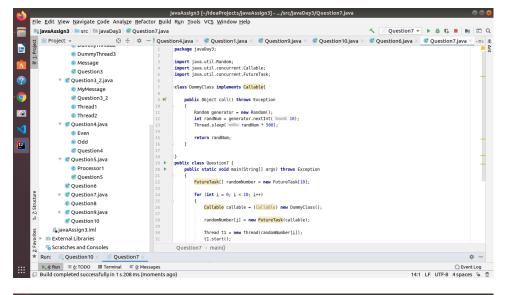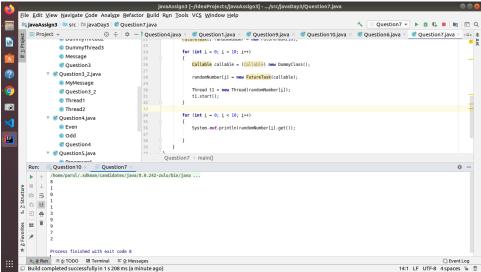
Question: 6:
Write a program to demonstrate sleep and join methods.

Question: 7:
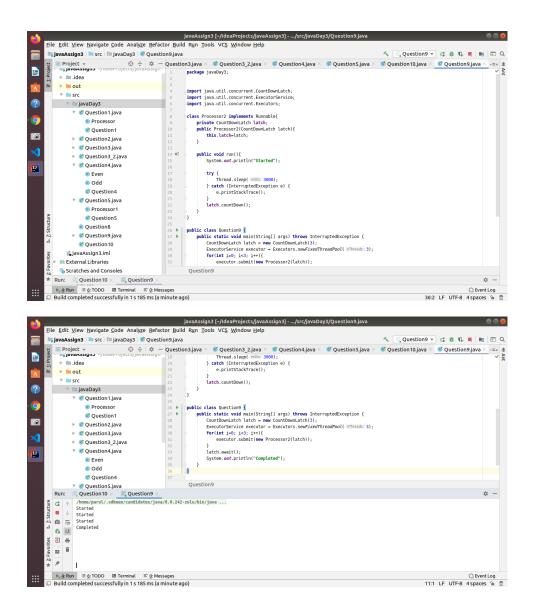Run a task with the help of callable and store it's result in the Future.

Question: 8:
Write a program to demonstrate the use of semaphore.

Question: 9:
Write a program to demonstrate the use of CountDownLatch.

Question: 10:

Write a program which creates deadlock between 2 threads.

```java
package javaDay3;

public class Question10 {
    public static void main(String[] args) {
        final String resource1 = "First Resource";
        final String resource2 = "Second Resource";
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized (resource1) {
                    System.out.println("Thread1--------->Resource 1");

                    try {
                        Thread.sleep( millis: 100);
                    } catch (Exception e) {

                    }

                    synchronized (resource2) {
                        System.out.println("Thread1----------->Resource 2");
                    }
                }
            }
        }) {
        };

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized (resource2) {
```

Question10 › main() › new Thread › new Runnable › run()

```java
            }
        }) {
        };

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized (resource2) {
                    System.out.println("Thread2----------->Resource 2");

                    try {
                        Thread.sleep( millis: 100);
                    } catch (Exception e) {

                    }

                    synchronized (resource1) {
                        System.out.println("Thread2----------->Resource 1");
                    }
                }
            }
        }) {
        };

        t1.start();
        t2.start();
    }
}
```

Question10 › main() › new Thread › new Runnable › run()

```java
            }
        }) {
        };

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized (resource2) {
                    System.out.println("Thread2----------->Resource 2");

                    try {
                        Thread.sleep( millis: 100);
                    } catch (Exception e) {

                    }

                    synchronized (resource1) {
                        System.out.println("Thread2----------->Resource 1");
                    }
```

Question10 › main() › new Thread › new Runnable › run()

Run:
```
/home/parul/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Thread1--------->Resource 1
Thread2----------->Resource 2
```

Build completed successfully in 1 s 172 ms (a minute ago)