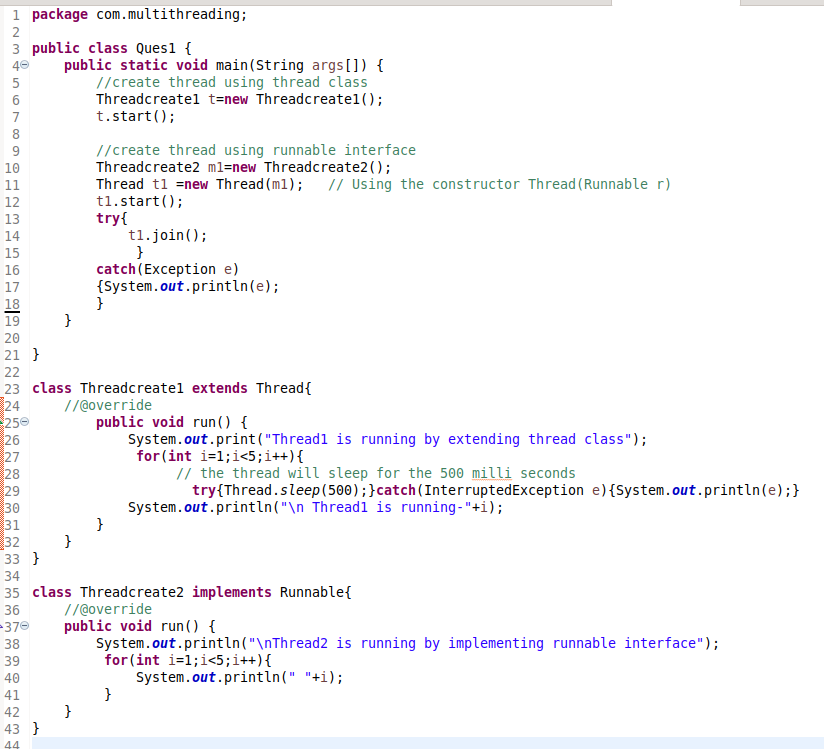
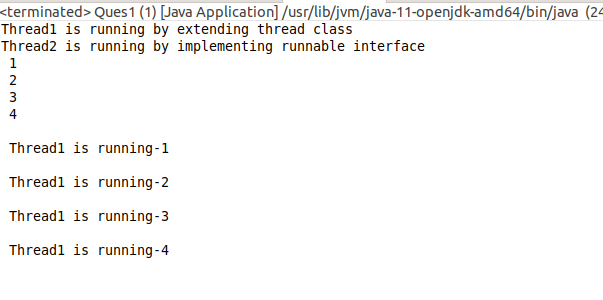
MULTITHREADING ASSIGNMENT HONEY ARORA

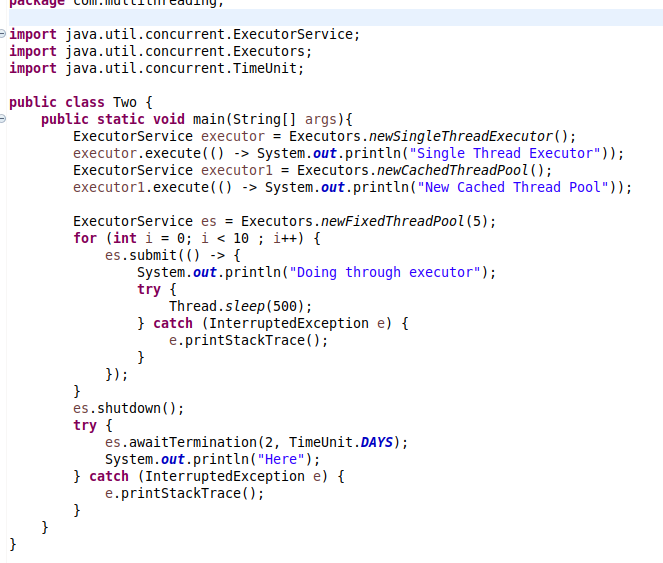
Trainee

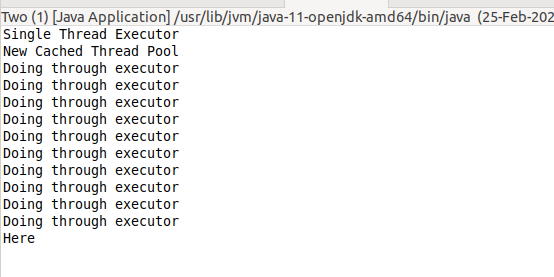
**QUES1.** **Create and Run a Thread using Runnable Interface and Thread class and show usage of sleep and join methods in the created threads.**

****

****

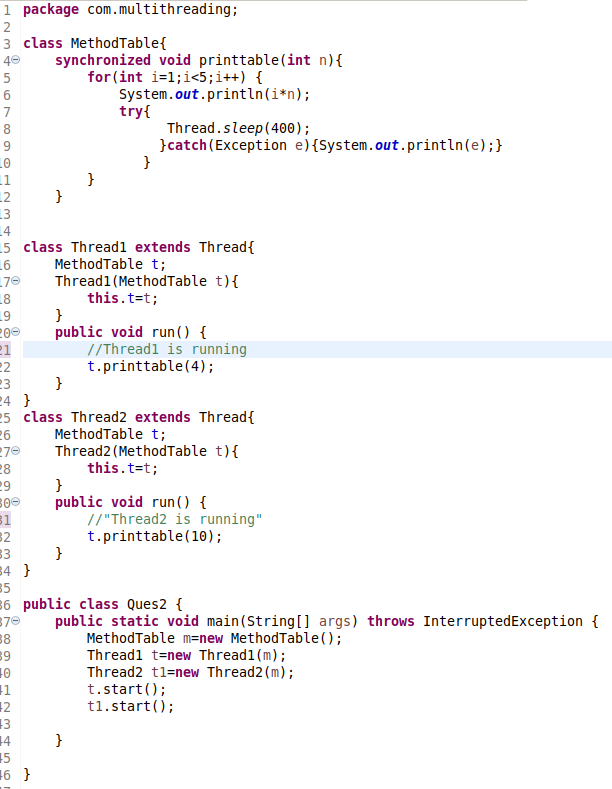
**QUES2. Use a singleThreadExecutor, newCachedThreadPool() and newFixedThreadPool() to submit a list of tasks and wait for completion of all tasks.**

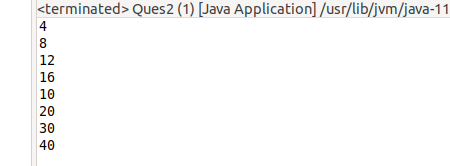
****

****

**QUES3. Use Synchronize method and synchronize block to enable synchronization between multiple threads trying to access method at same time.**

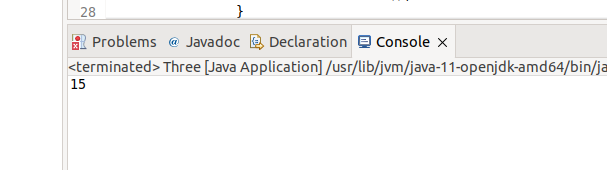
**Synchronize method:**

****

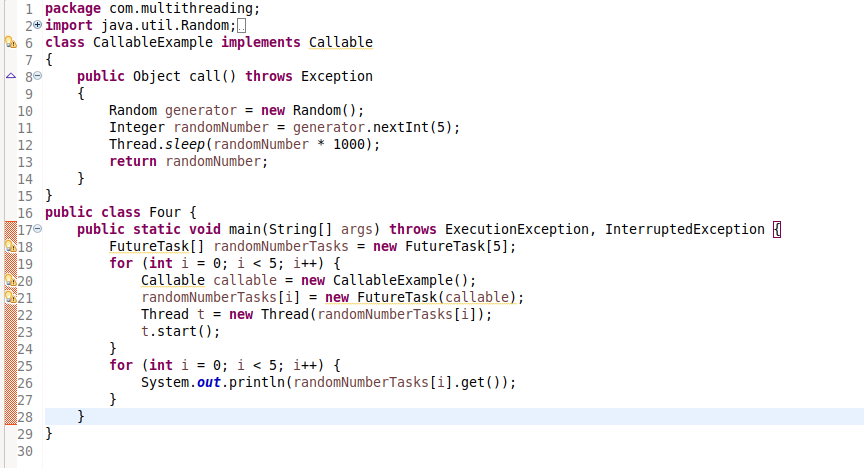
****

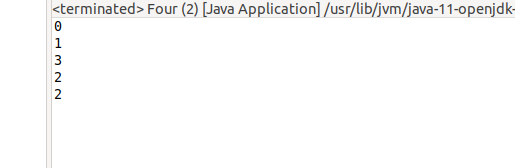
**Synchronize block**

****

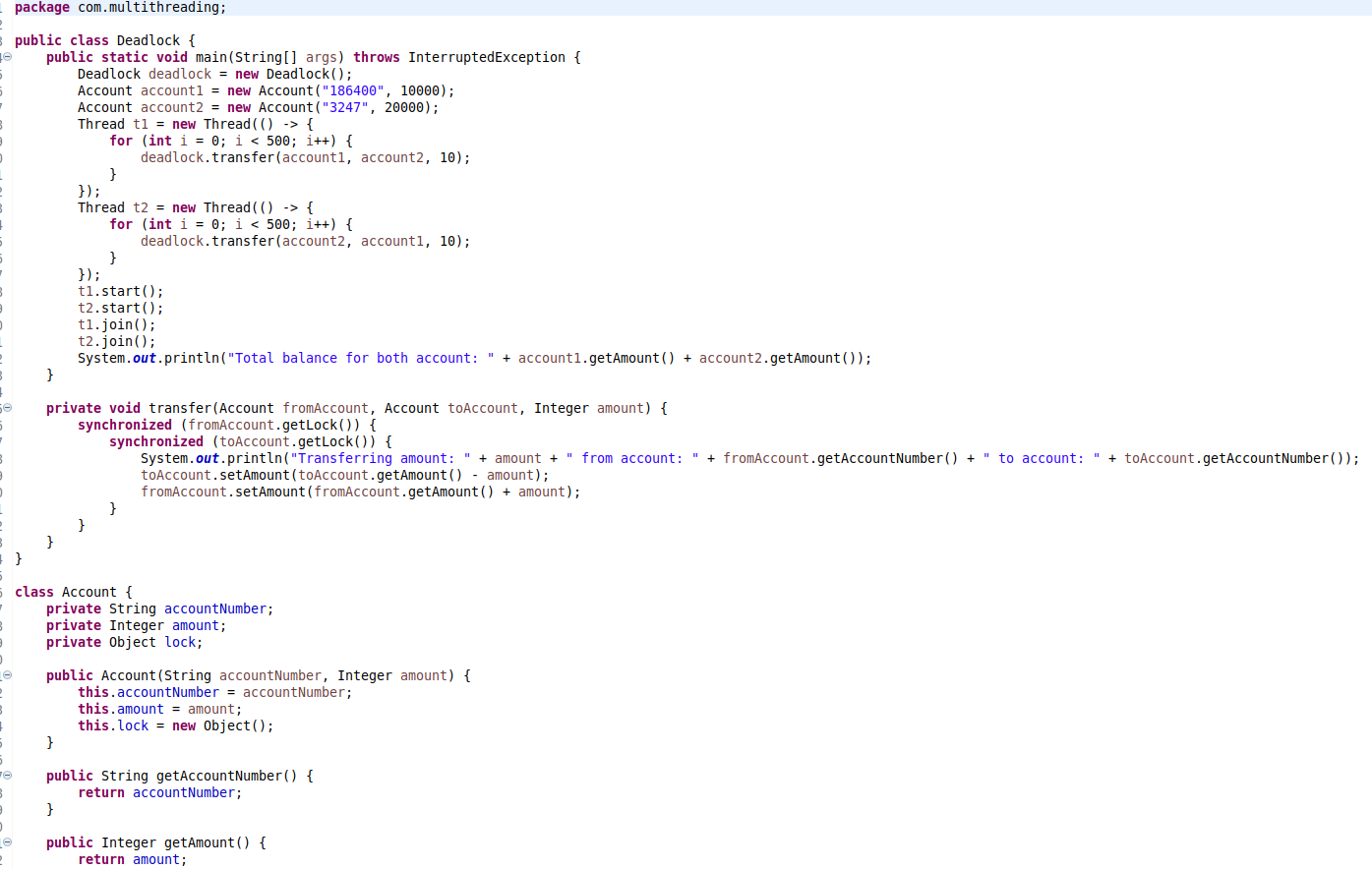
****

**QUES4. WAP to show usage of Callable and demonstrate how it is different from Runnable**

****

****

**QUES5. Write a program to simulate a deadlock in java and then improve the program to handle the deadlock using reentract lock.**

****

**(below pasted code)**

package com.multithreading;

public class Deadlock {

public static void main(String[] args) throws InterruptedException {

Deadlock deadlock = new Deadlock();

Account account1 = new Account("186400", 10000);

Account account2 = new Account("3247", 20000);

Thread t1 = new Thread(() -> {

for (int i = 0; i < 500; i++) {

deadlock.transfer(account1, account2, 10);

}

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 500; i++) {

deadlock.transfer(account2, account1, 10);

}

});

t1.start();

t2.start();

t1.join();

t2.join();

System.out.println("Total balance for both account: " + account1.getAmount() + account2.getAmount());

}

private void transfer(Account fromAccount, Account toAccount, Integer amount) {

synchronized (fromAccount.getLock()) {

synchronized (toAccount.getLock()) {

System.out.println("Transferring amount: " + amount + " from account: " + fromAccount.getAccountNumber() + " to account: " + toAccount.getAccountNumber());

toAccount.setAmount(toAccount.getAmount() - amount);

fromAccount.setAmount(fromAccount.getAmount() + amount);

}

}

}

}

class Account {

private String accountNumber;

private Integer amount;

private Object lock;

public Account(String accountNumber, Integer amount) {

this.accountNumber = accountNumber;

this.amount = amount;

this.lock = new Object();

}

public String getAccountNumber() {

return accountNumber;

}

public Integer getAmount() {

return amount;

}

public void setAmount(Integer amount) {

this.amount = amount;

}

public Object getLock() {

return lock;

}

}

