Java RMI Application

Students

Jack McGirl - 15421742

Meadhbh Fitzpatrick - 17348933

Testing

The core functionality of the program was testing using basic JUnit tests, e.g. getUsername(), getBalance() for the basic operations that would be enacted on an account.

We were unable to get the RMI calls working over JUnit tests so they were tested by hand, first to ensure correct operation, so an account was logged into, the balance checked and money was deposited and withdrawn and the resulting balance checked accordingly.

We then stepped through incorrect situations by passing incorrect details to the login function, as well as incorrect account numbers and session IDs to the deposit and withdraw functions, this ensured the code operated as expected and our exception handling worked also. They were also tested by passing strings where Int or BigDecimal was to be expected to ensure exceptions were thrown here too.

Finally we ran the same tests as above but with different delimiters for the input string.

Screenshots of Terminal

```
Jack@JACK-DESKTOP C:\Users\Jack\gitrepos\DistributedSystems\BankingRMI_App\bin
$ rmiregistry
```

```
Jack@JACK-DESKTOP C:\Users\Jack\gitrepos\DistributedSystems\BankingRMI_App\bin
$ java Bank
Account created. Username :JackMcGirl Account No. : 90
Account created. Username :MeadhbhFitzpatrick Account No. : 79
Server ready
```

```
Jack@JACK-DESKTOP C:\Users\Jack\gitrepos\DistributedSystems\BankingRMI_App\bin
$ java BankClient
Connected to Server
login,JackMcGirl,1234
Login Sucessful. Session ID: 3555
Please enter instructions in the form "instruction,accountNum,amount,sessionID "

Valid Instructions : deposit, withdraw, balance
balance,90,3555
Account Balance is: 10.5
withdraw,90,50,3555
balance,90,3555
Account Balance is: -39.5
deposit,90,150,3555
balance,90,3555
Account Balance is: 110.5
```

After 5 minutes pass

```
Session for user nullexpired at 22:28:28.086191200
```

We weren't able to figure out how to print the username into this message correctly.

Source Code

The source code can be found on Git here

https://github.com/DigitalSolitude/DistributedSystems/tree/main/BankingRMI_App/

Serverside

Exceptions

```
public class InvalidLogin extends Exception {


	private static final long serialVersionUID = -7599514687910701613L;

	public String username;


	public String getUsername() {

		return username;

	}

}
```

```
public class InvalidSession extends Exception {



	private static final long serialVersionUID = 9155299909449704444L;
	public String username;


	public String getUsername() {
		return username;
	}
}
```

Interfaces

```
import java.io.Serializable;
import java.rmi.Remote;
import java.time.LocalDate;
import java.util.List;


public interface IStatement extends Serializable, Remote {


public int getAccountnum();  // returns account number associated with this statement
public LocalDate getStartDate(); // returns start Date of Statement
public LocalDate getEndDate(); // returns end Date of Statement
public String getAccoutName(); // returns name of account holder
public List<Transaction> getTransactions(); // return list of transactions included in
this statement


}
import java.beans.Statement;
import java.math.BigDecimal;
import java.rmi.Remote;
import java.rmi.RemoteException;
```

```
import java.util.Date;


public interface IBank extends Remote {

	// The login method returns a token that is valid for some time period that must be passed to the other methods as a session identifier

		public long login(String username, String password) throws RemoteException, InvalidLogin, InvalidSession;

		public void deposit(int accountnum, BigDecimal amount, long sessionID) throws RemoteException, InvalidSession;

		public void withdraw(int accountnum, BigDecimal amount, long sessionID) throws RemoteException, InvalidSession;

		public BigDecimal getBalance(int accountnum, long sessionID) throws RemoteException, InvalidSession;

		public Statement getStatement(Date from, Date to, long sessionID) throws RemoteException, InvalidSession;


}
```

Core Classes - Bank

```
import java.beans.Statement;

import java.math.BigDecimal;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

import java.rmi.server.UnicastRemoteObject;


public class Bank extends UnicastRemoteObject implements IBank {

	private static List<Account> accounts = new ArrayList<Account>(); // users accounts
```

```
	private static List<Transaction> transactions = new ArrayList<Transaction>();
	private static Statement statement;
	Session session;



	public Bank() throws RemoteException
	{
		super();
	}



	public long login(String username, String password) throws RemoteException,
InvalidLogin, InvalidSession {
		try {
			for (Account i:accounts)
			{
				if (i.username.equals(username))
				{
					if (i.password.equals(password))
					{
						Session sesh = new Session(username);
						session = sesh;
						System.out.println("Login Sucessful. Session "
+ sesh.id + " is valid for 5 minutes.");
						return sesh.id;
					}
					System.out.println("Login Failed. Invalid Password " +
password);
					throw new InvalidLogin();
				}
				System.out.println("Login Failed. Invalid Username " +
username);
```

```
                        throw new InvalidLogin();
                }
                throw new InvalidLogin();
        }
        catch (InvalidLogin IL){
                System.out.println("Invalid Login for User: "
+IL.getUsername());
                }
        return session.id;
    }


    public void deposit(int accountnum, BigDecimal amount, long sessionID) throws
RemoteException, InvalidSession {
        try {
                session.sessionMonitor();
                if (sessionID != session.id) {
                        throw new InvalidSession();
                        }
                for (Account i:accounts) {
                        if (accountnum == i.accountNum)
                        {
                                transactions.add(new Transaction(amount,
LocalDate.now(), "Deposit"));
                                i.balance = i.balance.add(amount);
                                System.out.println(i.balance.toString());
                        }
                }
        }
        catch (InvalidSession IS) {
                System.out.println("Invalid Session for User: " +IS.getUsername());
        }
```

```
	}


	public void withdraw(int accountnum, BigDecimal amount, long sessionID) throws
RemoteException, InvalidSession {
		try {
			session.sessionMonitor();
			if (sessionID != session.id) {
				throw new InvalidSession();
				}
			for (Account i:accounts) {


				if (accountnum == i.accountNum)
				{
					transactions.add(new Transaction(amount,
LocalDate.now(), "Withdraw"));
					i.balance = i.balance.subtract(amount);
				}
			}
		}
		catch (InvalidSession IS) {
			System.out.println("Invalid Session for User: " +IS.getUsername());
		}
	}


	public BigDecimal getBalance(int accountnum, long sessionID) throws
RemoteException, InvalidSession {
		try {
			session.sessionMonitor();
			if (sessionID != session.id) {
				throw new InvalidSession();
				}
```

```
                for (Account i:accounts) {
                        if (accountnum == i.accountNum)
                        {
                                return i.balance;
                        }
                }
        }
        catch (InvalidSession IS) {
                System.out.println("Invalid Session for User: " +IS.getUsername());
        }
        return null;
    }


    public Statement getStatement(LocalDate from, LocalDate to, long sessionID) throws
RemoteException, InvalidSession {
        try {
                session.sessionMonitor();
                if (sessionID != session.id) {
                        throw new InvalidSession();
                        }
                for (Transaction t:transactions) {
                        if (t.date.isAfter(from)){
                                if (t.date.isBefore(to)) {
                                        System.out.println(t.toString());
                                }
                        }
                }
        }
                catch (InvalidSession IS) {
                        System.out.println("Invalid Session for User: "
+IS.getUsername());
```

```
                }
            return null;
        }


        public static void main(String args[]) throws Exception {
            // initialise Bank server - see sample code in the notes and online RMI
tutorials for details
            //Not used to working with BigDecimal (Or Java these days) so this is a little
awkward
            try {
                BigDecimal dec = new BigDecimal(10.50);
                Account jmg = new Account("JackMcGirl", "1234", dec);
                dec = dec.add(dec);
                Account mf = new Account("MeadhbhFitzpatrick", "2020", dec);
                accounts.add(jmg);
                accounts.add(mf);
                System.out.println("Account created. Username :" + jmg.username + "
Account No. : " + jmg.accountNum);
                System.out.println("Account created. Username :" + mf.username + "
Account No. : " + mf.accountNum);
              IBank bank = new Bank();
              Registry registry = LocateRegistry.createRegistry(2001);
              registry.rebind("Bank", bank);
              System.err.println("Server ready");
            } catch (Exception e) {
              System.err.println("Server exception: " + e.toString());
              e.printStackTrace();
            }
             }
}
```

Core Classes - Account

```
import java.math.BigDecimal;

import java.rmi.RemoteException;

import java.util.concurrent.ThreadLocalRandom;


public class Account {


	public String username;

	public String password;

	public int accountNum; // Random Start point

	public BigDecimal balance;


	public Account(String Username, String password, BigDecimal openingBalance) throws
RemoteException {

		username = Username;

		this.password = password;

		balance = openingBalance;

		accountNum = ThreadLocalRandom.current().nextInt(100);

	}


}
```

Core Classes - Session

```
import java.rmi.RemoteException;

import java.time.LocalTime;

import java.util.concurrent.ThreadLocalRandom;


public class Session {
```

```
	public long id;
	public String username;
	public LocalTime expireTime;
	public LocalTime currTime;
	public Boolean expired;

	// Creates a session from a passed Username
	public Session(String username) throws InvalidSession, RemoteException {
		setID();
		username = this.username;
		setTime();
		expired = false;
		sessionMonitor();
	}

	// Session Time to last 5 minutes
	private void setTime() {
		expireTime = LocalTime.now().plusMinutes(5);
		currTime = LocalTime.now();
	}

	// Generates a random Session ID between 0 and 10,000
	private void setID() {
		id = ThreadLocalRandom.current().nextLong(10000);
	}

	// Compares current time to the set expiry time, if the current time is after the
session time, throw an exception
```

```
	public void sessionMonitor() throws InvalidSession {
		try {
			currTime = LocalTime.now();
			if (currTime.isAfter(expireTime)) {
				expired = true;
				throw new InvalidSession();
			}
		}
		catch (InvalidSession IS){
			System.out.println("Session for user " +IS.getUsername()+ "expired
at " +expireTime.toString());
		}
	}
}
```

Core Classes - Transaction

```
import java.math.BigDecimal;
import java.time.LocalDate;
import java.io.Serializable;

public class Transaction implements Serializable {

	private static final long serialVersionUID = 8803216572303579376L;
	public BigDecimal amount;
	public LocalDate date;
	public String description;

	// Needs some accessor methods to return information about the transaction
```

```
	public Transaction(BigDecimal amount, LocalDate date, String description) {
		amount = this.amount;
		date = this.date;
		description = this.description;
	}

	public BigDecimal getAmount() {
		return amount;
	}

	public LocalDate getDate() {
	return date;
	}

	public void PrintTransaciton(Transaction t) {
		System.out.println("Transaction Details : Amount - " + t.amount.toString()
+ "Date - " + t.date.toString() + t.description);
	}
}
```

Core Classes - Statement

```
import java.util.List;
import java.time.LocalDate;

public class Statement implements IStatement  {

	private static final long serialVersionUID = -3393428484808591906L;
	private int accountNum;
```

```
    private LocalDate startDate, endDate;
    private String accountName;
  private List<Transaction> t;

    public int getAccountnum() {
        return accountNum;
    }

    public LocalDate getStartDate() {
        return startDate;
    }

    public LocalDate getEndDate() {
        return endDate;
    }

    public String getAccoutName() {
        return accountName;
    }

    public List<Transaction> getTransactions() {
        return t;
    }

}
```

Client Side - BankClient (Equivalent to ATM class)

```
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.Scanner;

public class BankClient {

	public static void main(String args[]) throws InvalidLogin, InvalidSession
	{
		try {
			IBank server = (IBank) Naming.lookup("//localhost:2001/Bank");
			String name;
			String password;
			BigDecimal balance;
			long sessionId;
			System.out.println("Connected to Server");
			Operations(server);
		} catch (MalformedURLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (RemoteException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (NotBoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
```

```
		}

	}


	public static void Operations(IBank server) throws RemoteException, InvalidLogin,
InvalidSession {

		Scanner in = new Scanner(System.in);

		String toSplit = in.nextLine();

		String[] instructions = toSplit.split(",");

		BigDecimal amount = null;

		if (instructions.length == 2) {

			amount = new BigDecimal(instructions[1]);

		}

		else {

			amount = new BigDecimal(instructions[2]);

		}

		if (instructions[0].equals("login")) {

			System.out.println("Login Sucessful. Session ID: " +
server.login(instructions[1], instructions[2]));

			System.out.println("Please enter instructions in the form
\"instruction,accountNum,amount,sessionID \"");

			System.out.println("Valid Instructions : deposit, withdraw,
balance");

			Operations(server);

		}

		else if (instructions[0].equals("deposit")) {

			server.deposit(Integer.parseInt(instructions[1]), amount,
Long.parseLong(instructions[3]));

			Operations(server);

		}

		else if (instructions[0].equals("withdraw")) {
```

```
                    server.withdraw(Integer.parseInt(instructions[1]), amount,
Long.parseLong(instructions[3]));

                    Operations(server);

             }

             else if (instructions[0].equals("balance")) {

                    System.out.println("Account Balance is: " +
server.getBalance(Integer.parseInt(instructions[1]),
Long.parseLong(instructions[2])));

                    Operations(server);

             }

             else {

                    System.out.println("Invalid Instruction: " + instructions[0]);

                    Operations(server);

             }

      }

}
```