```
package finalMiniProj;
//package com.me.mega;
import java.util.*;
import java.util.Date;
import java.sql.*;

//import com.me.miniproj.Block;
//import com.me.miniproj.Blockchain;

//import com.me.miniproj.Appointment1;
//import com.me.miniproj.Calculate;
//import com.me.miniproj.Customer;
//import com.me.miniproj.Doctorr;

import java.text.*;
import java.time.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

class Block {

  private int index;
  private long timestamp;
  private String hash;
  private String previousHash;
  private String data;
  private int nonce;

  public Block(int index, long timestamp, String previousHash, String data) {
    this.index = index;
    this.timestamp = timestamp;
```

```
  this.previousHash = previousHash;
  this.data = data;
  nonce = 0;
  hash = Block.calculateHash(this);
}

public int getIndex() {
  return index;
}

public long getTimestamp() {
  return timestamp;
}

public String getHash() {
  return hash;
}

public String getPreviousHash() {
  return previousHash;
}

public String getData() {
  return data;
}

public String str() {
  return index + timestamp + previousHash + data + nonce;
}

public String toString() {
```

```
  String builder = new String();
  builder=builder+"Block #"+index+" [previousHash : "+previousHash+", "+
  "timestamp : "+new Date(timestamp)+", "+"data : "+data+", "+
  "hash : "+hash+"]";
  return builder.toString();
}

public static String calculateHash(Block block) {
  if (block != null) {
    MessageDigest digest = null;

    try {
      digest = MessageDigest.getInstance("SHA-256");
    } catch (NoSuchAlgorithmException e) {
      return null;
    }

    String txt = block.str();
    final byte bytes[] = digest.digest(txt.getBytes());
     String builder = new String();

    for (final byte b : bytes) {
      String hex = Integer.toHexString(0xff & b);

      if (hex.length() == 1) {
        builder=builder+"0";
      }

      builder=builder+hex;
    }
```

```
    return builder.toString();
   }

   return null;
  }

  public void mineBlock(int difficulty) {
   nonce = 0;
          String k=null;
          for(int i=0;i<difficulty;i++)
                  k=k+"0";
   while (!getHash().substring(0,  difficulty).equals("0000")) {                    //You changed
    nonce++;
    hash = Block.calculateHash(this);
   }
  }

}

class Blockchain {

          private int difficulty;
          private ArrayList<Block> blocks;

          public Blockchain(int difficulty) {
           this.difficulty = difficulty;
           blocks = new ArrayList<Block>();
           // create the first block
           Block b = new Block(0, System.currentTimeMillis(), null, "First Block");
           b.mineBlock(difficulty);
           blocks.add(b);
```

```
}

public int getDifficulty() {
 return difficulty;
}

public Block latestBlock() {
 return blocks.get(blocks.size() - 1);
}

public Block newBlock(String data) {
 Block latestBlock = latestBlock();
 return new Block(latestBlock.getIndex() + 1, System.currentTimeMillis(),
    latestBlock.getHash(), data);
}

public void addBlock(Block b) {
 if (b != null) {
  b.mineBlock(difficulty);
  blocks.add(b);
 }
}

public boolean isFirstBlockValid() {
 Block firstBlock = blocks.get(0);

 if (firstBlock.getIndex() != 0) {
  return false;
 }

 if (firstBlock.getPreviousHash() != null) {
```

```
    return false;
  }

  if (firstBlock.getHash() == null ||
       !Block.calculateHash(firstBlock).equals(firstBlock.getHash())) {
    return false;
  }

  return true;
}

public boolean isValidNewBlock(Block newBlock, Block previousBlock) {
          // if(cal.amt!=cal.fees)
                    //  return false;
  if (newBlock != null  &&  previousBlock != null) {
    if (previousBlock.getIndex() + 1 != newBlock.getIndex()) {
      return false;
    }

    if (newBlock.getPreviousHash() == null  ||
              !newBlock.getPreviousHash().equals(previousBlock.getHash())) {
      return false;
    }

    if (newBlock.getHash() == null  ||
              !Block.calculateHash(newBlock).equals(newBlock.getHash())) {
      return false;
    }

    return true;
  }
```

```
  return false;
}

public boolean isBlockChainValid() {
  if (!isFirstBlockValid()) {
    return false;
  }

  for (int i = 1; i < blocks.size(); i++) {
    Block currentBlock = blocks.get(i);
    Block previousBlock = blocks.get(i - 1);

    if (!isValidNewBlock(currentBlock, previousBlock)) {
      return false;
    }
  }

  return true;
}

public String toString1() {
  String builder=new String();

  for (Block block : blocks) {
    builder=builder+block+"\n";
  }

  return builder.toString();
}
```

```
        }



class Customer{
        String custid,custname;                         //User admin later
        int age;
        public void setCust(){
                Scanner sc=new Scanner(System.in);
                System.out.println("Cutomer Id:");
                custid=sc.next();
                System.out.println("Cutomer Name:");
                custname=sc.next();
                System.out.println("Age");
                age=sc.nextInt();
        }
        void display(){
                System.out.println("----------Details----------");
                System.out.println("Customer Id: "+custid+" Name: "+custname+" Age: "+age);
        }
}

class Doctorr extends Customer{                                         ///extend to
customer
        String docname,docid;
        String specialisation;
        int fees;
        void setDoc(String id,String name,String spc,int f){
                docid=id;
                docname=name;
```

```
		specialisation=spc;
		fees=f;
	}
	void dispDoc(){
		System.out.println("Doctor Details:");
		System.out.println("ID:"+docid+"Name: "+docname+" Fees: "+fees);
	}
}

class Calculate extends Doctorr{
	double tax,total;
	int amt;
	void tax_cal(){
		System.out.println("Enter your mode of payment\n1-Creadit Card   2-Debit Card   3-
Net Banking");
		Scanner sc=new Scanner(System.in);
		int opt=sc.nextInt();
		if(opt==1)
			tax=1.5;
		else if(opt==2)
			tax=1.2;
		else if(opt==3)
			tax=0.8;
		else
			System.out.println("Invalid Option");
	}
	void calc(){
		total=fees+tax*fees;
	}
	void dispBill(){
		System.out.println("--------------------BILL-----------------------");
```

```
		System.out.println("Doctor_Name\t\t\tNumber_of_hours\t\tFees\t\tTax\t\t\tTotal");

		System.out.println(docname+"\t\t\t\t\t"+2+"\t\t\t"+fees+"\t\t\t\t"+tax+"\t\t\t"+total);
			//ABC is bullshit
		}
}

class Appointment1 extends Customer{
		double appno;
		String strDate,strTime;
		void setApp(){
			appno=Math.random()*20;
			Date date = new Date();
		  //String strDateFormat = "hh:mm:ss a";
		  SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
		  strDate= formatter.format(date);
		  Calendar cal = Calendar. getInstance();
		  date=cal. getTime();
		  DateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");
		  strTime=dateFormat. format(date);
		}
		void dispApp(){
			System.out.println("Your appointment has been scheduled for "+strDate+" at
"+strTime);
		}
}

public class MiniProjFinal {
		public static void main(String[] args) {
			// TODO Auto-generated method stub
			int flag1=0,flag2=0,k=1;
```

```
		String id;
		Scanner sc=new Scanner(System.in);
		/*String docname[]={"Katniss","Bri","Ash"};
		String docid[]={"D111","D222","D333"};
		String spc[]={"ENT","Ortho","Cardio"};
		int fee[]={750,2300,2500};*/
		System.out.println("\nWelcome!\n");
		System.out.println("Please enter your details to get started...");
		Customer cus=new Customer();
		cus.setCust();					//Do this using Calculate obj
		cus.display();
	//	Doctorr doc=new Doctorr();				//Make this an array....DONE....Not needed
		Calculate cal=new Calculate();

		try{
			Connection conn=DriverManager.getConnection("jdbc:mysql://localhost/sem6","root","");
			System.out.println("Connected to Database");

		System.out.println("List of categories available");		//Execute query
		Statement stmt1=conn.createStatement();
		ResultSet rs=stmt1.executeQuery("select distinct spc from doctor" );
		while(rs.next())
			System.out.println(rs.getString(1));		//DONE!
		System.out.println("Choose your category");			//Category WTF?
		String cat=sc.next();
		Statement stmt2=conn.createStatement();
		ResultSet rs1=stmt2.executeQuery("select * from doctor where 'spc'="+cat );
	//NOT HAPPENING
		System.out.println("Here are the list of doctors with the chosen specialisation");
		System.out.println("ID	Name	Specialisation	Fees");
```

```
                while(rs.next())
                        System.out.println(rs1.getString(1)+"    "+rs1.getString(2)+"
"+rs1.getString(3)+"    "+rs1.getInt(4));



                //System.out.println("Here are the list of doctors with the chosen specialisation");
                //System.out.println("  ID      Name        Specialisation        Fees");
                //for(int i=0;i<3;i++)
                        //System.out.println(" "+docid[i]+"    "+docname[i]+"        "+spc[i]+"
"+fee[i]);



                System.out.println("Here are the list of doctors with the chosen specialisation");
                System.out.println("  ID      Name        Specialisation        Fees");
                for(int i=0;i<3;i++){
                        if(cat.equals(spc[i])){

                                //for(int m=0;m<3;m++){
                                        //if(cat.equals(spc[m]))
                                                System.out.println(" "+docid[i]+"    "+docname[i]+"
"+spc[i]+"       "+fee[i]);
                                //}
                        }
                }
                                System.out.println("Enter the ID of the doctor of your choice");
                                id=sc.next();
                                int i;
                                for(i=0;i<3;i++){
                                        if(docid[i].equals(id)){                //Doesnt specify spc!!!
                                                cal.setDoc(docid[i],docname[i],spc[i],fee[i]);
        //Null Pointer Exception
```

```
                                        break;
                                }
                        }
                        //while(k==1){
                                cal.dispDoc();
                                //for(int j=0;j<3;j++){
                                        //if(docid[i].equals(id)){
        //....CHECK....basically no fails now
                                                System.out.println("Appointment booked
with Dr. "+cal.docname);                //Update this in the appointment class
                                                Appointment1 app=new Appointment1();
                                                app.setApp();
                                                app.dispApp();
                                                flag1=1;
                                                //break;
                                //}
                                //}
                                if(flag1==0){
                                        System.out.println("Incorrect Id");
                                        //System.out.println("Press 1 to book another
appointment 0 to exit");
                                        //k=sc.nextInt();
                                //}
                        }
                        flag2=1;
                        //break;
                //}
                if(flag2==0){
                        System.out.println("Appointments are currently unavailable");
                        System.exit(0);
                }
                //Calculate cal=new Calculate();
```

```
                System.out.println("\n\nAmount to be paid= "+cal.fees);

                cal.tax_cal();                          //All that payment
                cal.calc();
                System.out.println("\nEnter the payment amount");
                cal.amt=sc.nextInt();
                //Put it all in 1...

                cal.dispBill();
                Blockchain blockchain = new Blockchain(4);
            blockchain.addBlock(blockchain.newBlock("Block 1 Bitcoin"));
            blockchain.addBlock(blockchain.newBlock("Block for fees to be paid"));
            blockchain.addBlock(blockchain.newBlock("Block for amount being paid"));

            if(cal.amt!=cal.fees) {
                System.out.println("Blockchain Valid ? False");
                System.out.println("Invalid Transaction!!");
                System.out.println("Appointment cancelled\nRestart to book another
appointment");
                System.exit(0);
            }

            System.out.println("Blockchain valid ? " + blockchain.isBlockChainValid());
            System.out.println(blockchain);
                }
                catch(Exception e){
                        System.out.println(e);
                }

        }
```

}