

**THE UNIVERSITY OF HONG KONG****FACULTY OF ENGINEERING  
DEPARTMENT OF COMPUTER SCIENCE****COMP2396/CSIS0396: Object-Oriented Programming and Java****Date: December 18, 2014****Time: 2:30pm-5:30pm**

*Only approved calculators as announced by the Examinations Secretary can be used in this examination. It is candidates' responsibility to ensure that their calculator operates satisfactorily, and candidates must record the name and type of the calculator used on the front page of the examination script.*

Brand: \_\_\_\_\_

Model: \_\_\_\_\_

**Write your University No. at the top of each page****The examination is THREE hours.****Answer ALL questions in the space provided.**

Question No.	Mark
1 (18%)	
2 (20%)	
3 (10%)	
4 (10%)	
5 (10%)	
6 (12%)	
7 (20%)	
<b>Total (100%)</b>	

**QUESTION 1 (18%)**

For each of the following Java programs, determine if it can compile without error(s) or not.

1. If it can compile without error, what is the output?
2. If it compiles with error(s)
  - a. Write down the problem(s).
  - b. How do you propose to fix the problem(s)?
  - c. What is the output of the fixed program?

(a)

```
class Student {
    String name;
    int id;
}

public class CSClass {
    public static void main( String[] args) {
        Student[] cs0396 = new Student[2];

        cs0396[0].name = "Kevin";
        cs0396[0].id = 12345;

        cs0396[1].name = "Haoyuan";
        cs0396[1].id = 67890;

        for (int i=0; i<2; ++i) {
            System.out.println( cs0396[i].name + " with id "
                               + cs0396[i].id);
        }
    }
}
```

(b)

```
class Animal {  
    public Animal() {  
        System.out.println("Animal()");  
    }  
    private String toString() {  
        return "Animal";  
    }  
    public static void main(String args[]) {  
        Animal a = new Animal();  
        System.out.println(a.toString());  
    }  
}
```

(c)

```
class Rectangle {  
    double width, height;  
    public Rectangle(double w, double h) {  
        width = w; height = h;  
        System.out.println("Rectangle()");  
    }  
}  
class Square extends Rectangle {  
    double size;  
    public Square(double s) {  
        size = s;  
        System.out.println("Square()");  
    }  
}  
public class Shape {  
    public static void main(String[] args) {  
        Square sq = new Square(1.0);  
        Rectangle re = new Rectangle(1.0, 2.0);  
    }  
}
```

(d)

```
// System.exit(0) terminates a program

public class HelloGoodbye {
    public static void main( String[] args) {
        try {
            System.out.println("Hello World");
            System.exit(0);
        } finally {
            System.out.println("Goodbye World");
        }
    }
}
```

(e)

```
class NonDigitException extends Exception {}

public class Try {
    static void check(final int d) {
        if ( d<'0' || d>'9' )
            throw new NonDigitException();
    }

    public static void main(String[] args) {
        int d = 11;
        try {
            check(d);
        } catch (NonDigitException e) {
            e.printStackTrace();
        }
    }
}
```

(f)

```
interface I {}  
class A {  
    A() {  
        System.out.print("A: " + (this instanceof I) + " ");  
    }  
    public static void main(String args[]) {  
        new C();  
    }  
}  
  
class B extends A implements I {  
    B() {  
        System.out.print("B: " + (this instanceof I) + " ");  
    }  
}  
  
class C extends B {  
    C() {  
        System.out.print("C: " + (this instanceof I) + " ");  
    }  
}
```

---

**QUESTION 2 (20%)**

Multiple choice questions. Circle the selected choices.

**QUESTION 3 (10%)**

Consider the following program

```
import java.io.*;

class Teacher implements Serializable {}

class GuestTeacher extends Teacher {}

public class HeadFirstTeacher {

    public static void main( String[] args) {
        Integer[] quotas = {7, 8, 9, 10};
        Teacher t1 = new Teacher();
        Teacher t2 = new GuestTeacher();
        Teacher t3 = new GuestTeacher();
        Teacher t4 = new Teacher();
        Teacher t5 = new Teacher();
        Integer[] duplicateQuotas = quotas;

        t1 = t4;
        t4 = null;
        t2 = t3;
        Teacher[] teacherAssignments = { t1, t5};
        manage(t2);
        // count number of objects here
    }

    static void manage(Teacher t) {
        GuestTeacher ta = new GuestTeacher();
        Teacher tb = t;
    }
}
```

For each class used in the above program, give the number of objects that have been created at the end of the execution. In addition, give the number of the objects abandoned.

Classes	No. of objects created	No. of objects abandoned
Integer	4	0

**QUESTION 4 (10%)**

Consider the following program

```
class PostGrad {
    int credits = 12;
    String name = "PostGraduate Student";
    PostGrad() { }
    PostGrad(String n) { name = n; }
    public String toString() {
        return "I am "+name+" and I need "+credits+"credits";
    }
}

class MSc {
    int credits = 24;
    String name = "MSc Student";
    MSc() { }
    MSc(String n) { name = n; }
    public String toString() {
        return "I am "+name+" and I need "+credits+"credits";
    }
}

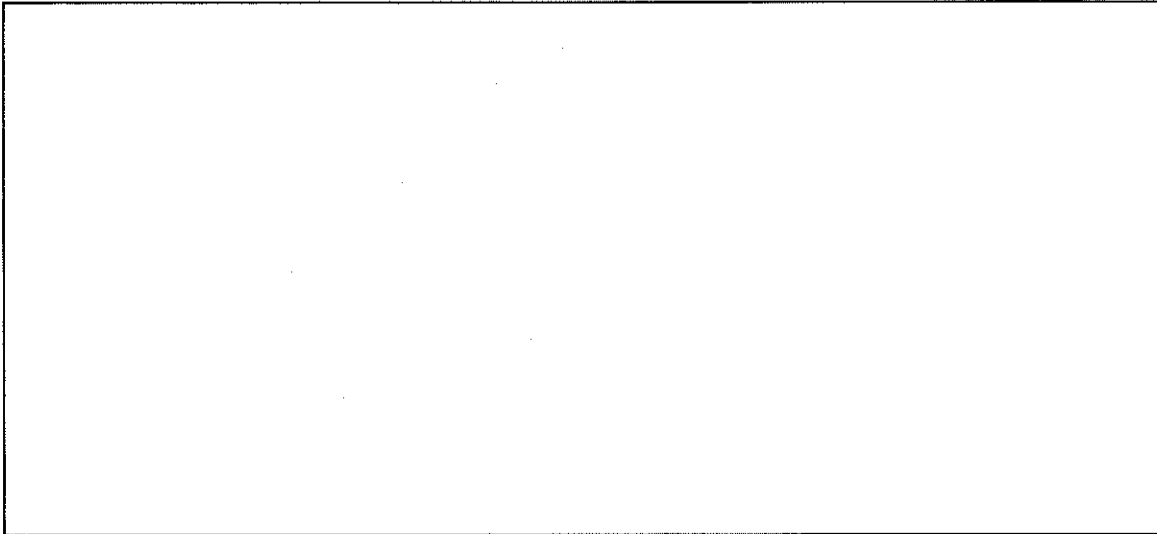
class PhD {
    int credits = 30;
    String name = "PhD Student";
    PhD() { }
    PhD(String n) { name = n; }
    public String toString() {
        return "I am "+name+" and I need "+credits+"credits";
    }
}

public class Test {
    public static void main( String[] args) {
        MSc m = new MSc("Alice");
        PhD p = new PhD("Bob");
        PostGrad pg1 = m;
        PostGrad pg2 = p;

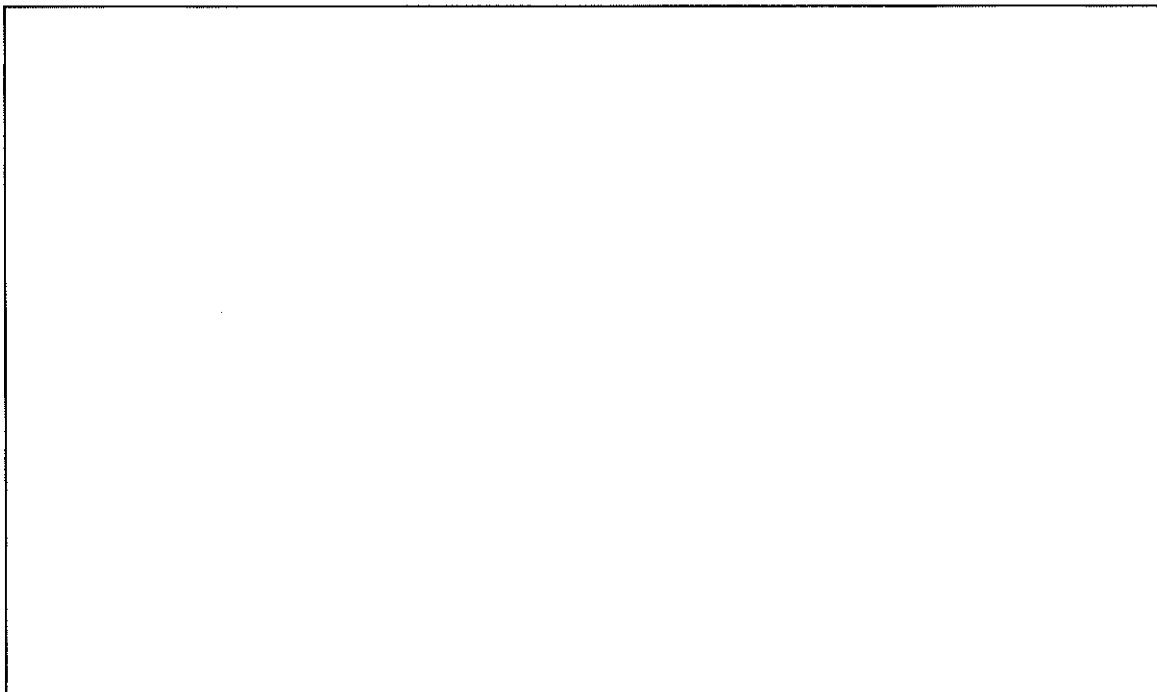
        System.out.println(m+"\n"+p+"\n"+pg1+"\n"+pg2);
        System.out.println("I am "+pg1.name+"and I need "+pg1.credits+ "credits");
        System.out.println("I am "+pg2.name+"and I need "+pg2.credits+ "credits");
        System.out.println("I am "+m.name+"and I need "+m.credits+ "credits");
        System.out.println("I am "+p.name+"and I need "+p.credits+ "credits");
    }
}
```



a) What is the output when the above program is executed?

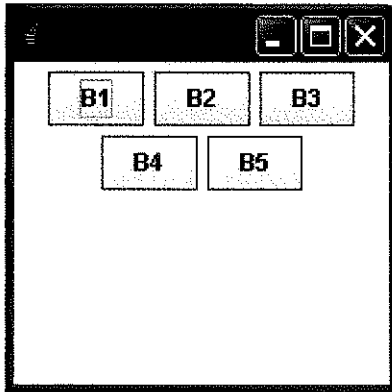


b) Explain your output in a) with respect to the language features of Java.

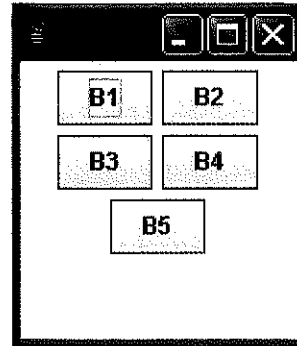




- b) Modify the above code fragment using the identical set of Swing components to produce the following screen display. Describe the layout policy that has been used in the Java code fragment.

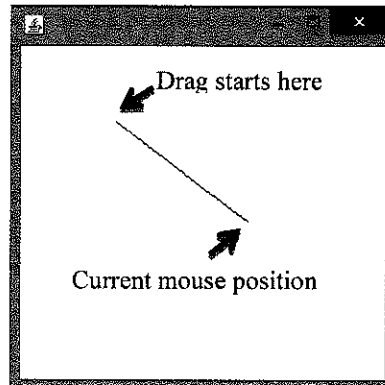
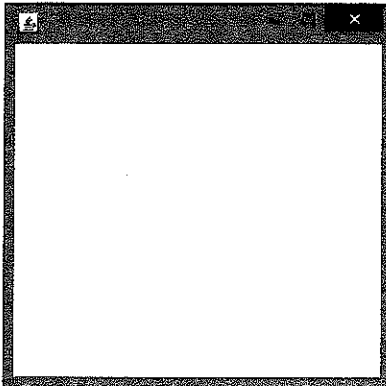


After resizing



**QUESTION 6 (12%)**

The following shows a Java program that displays a white rectangle (the left window). Re-write the program such that a line appears when a user drags on the white area. The line should connect the starting point and the current point of the dragging (the right window). Some probably useful APIs are also given. (The *MouseListener* and the *MouseMotionListener* is an interface for receiving mouse events, e.g., press, release, click, drag, etc. A *JFrame* can act as a source and a listener of a *MouseEvent* and a *MouseMotionEvent*.)



```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class Mouse extends JFrame {

    public static void main( String[] args) {
        (new Mouse()).go();
    }

    public void go() {
        setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);

        MyPanel p = new MyPanel();
        getContentPane().add( p);
        setSize( 300, 300);
        setVisible( true);
    }

    class MyPanel extends JPanel {
        public void paintComponent( Graphics g) {
            g.setColor( Color.white);
            g.fillRect( 0, 0, 300, 300);
        }
    }
}
```

**Methods in Interface MouseListener**

Modifier and Type	Method and Description
void	<u><b>mouseClicked(MouseEvent e)</b></u> Invoked when the mouse button has been clicked (pressed and released) on a component.
void	<u><b>mouseEntered(MouseEvent e)</b></u> Invoked when the mouse enters a component.
void	<u><b>mouseExited(MouseEvent e)</b></u> Invoked when the mouse exits a component.
void	<u><b>mousePressed(MouseEvent e)</b></u> Invoked when a mouse button has been pressed on a component.
void	<u><b>mouseReleased(MouseEvent e)</b></u> Invoked when a mouse button has been released on a component.

**Methods in Interface MouseListener**

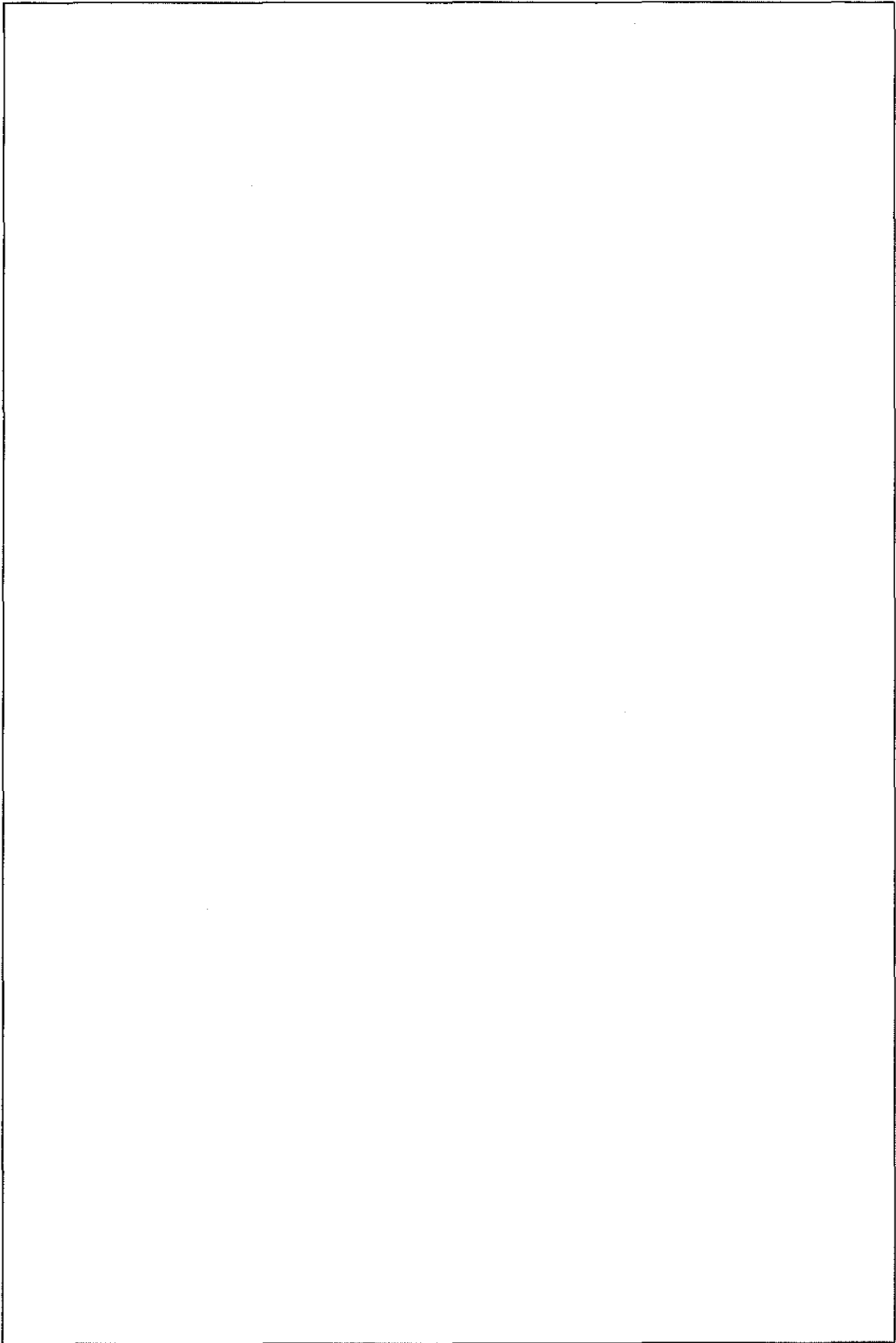
Modifier and Type	Method and Description
void	<u><b>mouseDragged(MouseEvent e)</b></u> Invoked when a mouse button is pressed on a component and then dragged.
void	<u><b>mouseMoved(MouseEvent e)</b></u> Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

**Some methods in class Graphics**

Modifier and Type	Method and Description
abstract void	<u><b>drawLine(int x1, int y1, int x2, int y2)</b></u> Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.
abstract void	<u><b>fillRect(int x, int y, int width, int height)</b></u> Fills the specified rectangle.
abstract void	<u><b>setColor(Color c)</b></u> Sets this graphics context's current color to the specified color.

**Some methods in class MouseEvent**

Modifier and Type	Method and Description
int	<u><b>getX()</b></u> Returns the horizontal x position of the event relative to the source component.
int	<u><b>getY()</b></u> Returns the vertical y position of the event relative to the source component.



**QUESTION 7. (20%)**

Consider a user registration system for an Internet forum. Each user of the forum has a unique username. The user information is stored in a text file “userinfo.txt” and the username and password is separated by “:”. It is assumed that “:” does not appear in the username and the password. Following is part of the “userinfo.txt” file:

```
Kevin Lam:wonderful
Haoyuan:hksars
KPC: 0396
...
```

The following Java classes are part of the user registration system. Class User is used to store the username and password of a registered user. All User objects are stored in a HashMap using the username as the key.

```
public class User implements Serializable
{
    String username;
    String password;

    public User(String n, String p) {
        username = n;
        password = p;
    }
}
public class UserList
{
    HashMap<String,User> users = new HashMap<String,User>();

    public UserList() { }

    public void getUserList() {
        try {
            File f = new File("userinfo.txt");
            BufferedReader br = new BufferedReader(new FileReader(f));
            String line = null;
            while ( ( line = br.readLine() ) != null ) addUser(line);
        }
        catch (Exception e) { }
    }

    void addUser(String line) {
        String[] toks = line.split(":");
        users.put(toks[0],new User(toks[0],toks[1]));
    }

    public boolean checkUser(String nm) {
        if ( users.containsKey(nm) ) return true;
        else return false;
    }
}
```

```

        public Boolean checkPassword(String nm, String pwd) {
            User u = users.get(nm);
            if ( u == null ) return false;
            if ( pwd.equals(u.password) ) return true;
            else return false;
        }
    }
}

```

The class `UserRegisterServer` provides user account management functions for the forum. The `UserRegisterServer` listens to port 5000 for incoming requests and provides the following services to the client:

- `checkusername`: check if a username has been registered or not
- `validatepassword`: validate the password of the user

The request from the client is stored in a `Message` object with the following definition:

```

public class Message
{
    String command;
    String username;
    String password;
    String result;

    public Message(String c, String n, String p) {
        command = c; username = n; password = p;
    }
}

```

The code fragment of `UserRegisterServer` that listens to incoming request and processes the request is as follows:

```

public class UserRegisterServer {

    UserList userList;

    public void go() {

        userList = new UserList();
        userList.getUserList();

        try {
            ServerSocket ss = new ServerSocket(5000);

            while (true) {
                Socket s = ss.accept();
                InputStream is = s.getInputStream();
                ObjectInputStream ois = new ObjectInputStream(is);
                OutputStream os = s.getOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(os);
            }
        }
    }
}

```



```

        Message m = (Message) ois.readObject();

        if ( m.command.equals("checkusername") )
        {
            boolean user_exist = userList.checkUser(m.username);
            if ( user_exist) m.result = "true";
            else m.result = "false";
            oos.writeObject(m);
        }
        else if ( m.command.equals("validatepassword") )
        {
            boolean validate = userList.checkPassword(m.username,
                                                        m.password);

            if ( validate ) m.result = "true";
            else m.result = "false";
            oos.writeObject(m);
        }
        else /* Add other commands here */
        {
            m.result = "no such command";
            oos.writeObject(m);
        }
    }

} catch (Exception e) { e.printStackTrace(); }

}

public static void main(String[] args)
{
    UserRegisterServer urs = new UserRegisterServer();
    urs.go();
}
}

```

The following is the Java code that is used by the client to submit a request to the UserRegisterServer.

```

public class UserRegisterClient {

    String command;
    String username;
    String password;
    String result;

    public static void main(String[] args) {

        UserRegisterClient urc = new UserRegisterClient();
        urc.go();
    }

    public void go ()
    {

```

```

try {
    Socket s = new Socket("userregserver.hk",5000);
    OutputStream os = s.getOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(os);
    InputStream is = s.getInputStream();
    ObjectInputStream ois = new ObjectInputStream(is);

    // get command from user, with username and password if needed
    ...

    Message m = new Message(command,username,password);
    oos.writeObject(m);
    m = (Message) ois.readObject();
    result = m.result;
    // process the result
    ...

} catch ( Exception e ) { e.printStackTrace(); }
}

```

- a) Assume that you are going to add the following new service to the UserRegisterServer:
- addusername: add a user to the **userlist** with username and password in the message request. If the username does not exist in the userlist, the addition is successful, put "success" in the result of the message; otherwise, put "fail" in the result of the message.

Insert code to process the new command "addusername" to the line marked with "Add other commands here". You may have to add new methods and modify existing methods in the class UserList (part b).

```

}
else
{
    m.result = "no such command";
    oos.writeObject(m);
}

```

- b) Modify the UserRegisterServer so that a new thread is started to handle each request from the client.

- c) Based on your revision to the UserRegisterServer in b), do you need to modify class UserList? If yes, please provide the necessary changes.

**\*\*\* E N D \*\*\***