Commencé le	mardi 29 novembre 2022, 08:05
État	Terminé
Terminé le	mardi 29 novembre 2022, 08:55
Temps mis	50 min 1 s
Note	**16,00** sur 21,00 (**76,19**%)

Question 1

Correct

Note de 1,00 sur 1,00

What is the output?

```
int a;
int b;
b = 31;
a = b;
b = 61;
System.out.println(a);
```

Réponse : 31 ✔

La réponse correcte est : 31

Question 2

Correct

Note de 1,00 sur 1,00

Soient les classes :

```
public abstract class Ant {
     public void sophia() {
          System.out.println("Blah");
     }
}

public abstract class Ipo extends Ant {
}

public class Lis extends Ipo {
}
```

Lesquelles des expressions suivantes **sont légales** (elles compilent et elles s'exécutent) :

Which of the following expressions are legal (they compile and they run):

Veuillez choisir au moins une réponse.

- `Lis place = new Ipo();`
- Lis place = new Ant();
- Lis place = (Lis) new Ipo();
- `Ipo place2 = new Lis();` ✔ Compile - Lis is concret so can be instantiated.
- Ant place = new Lis(); ✔ Compile - `Lis` is concret so can be instantiated.

Les réponses correctes sont : Ant place = new Lis();,

```
Ipo place2 = new Lis();
```

Question 3

Correct

Note de 1,00 sur 1,00

MAX_VALUE est un nom conventionnel Java pour :
package, class, instance method, variable, constante, rien (de ce qui précède).

Vous pouvez répondre en n'écrivant que les 2 première lettre de la réponse, exemple 'pa' pour 'package'.

MAX_VALUE is a conventional Java name for package, class, method, variable, constant, or nothing (of the above).

You can answer by writing only the first 2 letters of the answer, for example, 'pa' for 'package', 'ri' for nothing (rien).

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 # MAX_VALUE est un nom conventionnel Java pour :
2 co
```

	Got	Expected	Mark	
✔	co	constant	1	✔

Tous les tests ont été réussis ! ✔

Correct

Note pour cet envoi : 1,00/1,00.

Question 4

Correct

Note de 5,00 sur 5,00

The English version is below.

Vous allez développer un petit système pour envoyer et lire du courrier. Il y a un *serveur de courrier* ; les clients peuvent envoyer du courrier au serveur pour d'autres clients, et peuvent lire le courrier qui leur est envoyé par le serveur. Le système comporte deux classes qui ont déjà été développées par votre entreprise :

- ***MailItem*** représente un élément de courrier à échanger entre les clients et le serveur ;
- ***MailServer*** est le serveur qui gère les éléments de courrier entre les clients.

- **Téléchargez-les depuis le répertoire qui vous est donné**

Il y a une troisième classe qui est manquante parce que votre entreprise n'avait personne ayant les compétences pour l'écrire. Et c'est pourquoi vous avez été engagé, pour développer la classe ***MailClient***.

Une structure squelette pour la classe ***MailClient*** vous est donnée ; vous devez implémenter ses méthodes.

Une fois que vous avez terminé le développement de ***MailClient.***

Copiez **tout le code de la classe *MailClient* uniquement**, collez-le dans la boîte de réponse et vérifiez la réponse.
Ne modifiez aucune des autres classes (MailServer et MailItem) ; ne soumettez aucune de ces autres classes.

English version

You are going to develop a small system for sending and reading mail. There is a mail server; clients can send mail to the server for other clients and can read mail sent to them by the server. The system has two classes that your company has already developed:

1. MailItem represents a mail item to be exchanged between the clients and the server;
2. MailServer is the server that manages the mail items between the clients.

- **Download them from the directory given to you**

A third class is missing because your company needed someone with the skills to write it. And that's why you were hired to develop the **MailClient class.**

You are given a skeleton structure for the MailClient class; you must implement its methods.

Once you have finished developing MailClient:

Copy all the code of this class only, paste it into the response box and check the response.

Do not modify any other classes (MailServer and MailItem); do not submit any other classes.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  package test3.mailsystem;
2  /**
3   * A class to model a simple email client. The client is run by a particular
4   * user, and sends and retrieves mail via a particular server.
5   *
6   * @author David J. Barnes and Michael Kolling
7   * @author MBF
8   */
9
10 public class MailClient implements MailClientInterface {
11     //The server used for sending and receiving.
12     private MailServer server;
13
14     //The user  running this client.
15     private String user;
16
17     /**
18      * Create a mail client run by user and dependent on the the given server.
19      */
20     MailClient(MailServer server, String user) {
21         this.server = server;
22         this.user = user;
23     }
```

```
24
25      /**
26       * Return the next mail item (if any) for this user presents on the linked server
27       * else return null;
28       */
29      @Override
30      public MailItem getNextMailItem() {
31          return server.getNextMailItem(user);
32      }
33
34      /**
35       * Returns the next mail item (if any) for this user as a string
36       */
37      @Override
38      public String nextMailItemToString() {
39          MailItem item = getNextMailItem();
40          if (item == null) {
41              return "No new mail.";
42          } else {
43              return item.toString();
44          }
45      }
46
47      /**
48       * Send the given message to the given recipient via the attached mail
49       * server.
50       *
51       * @param to      The intended recipient.
52       * @param message The text of the message to be sent.
```

	Test	Résultat attendu	Résultat obtenu	

✔	server = new MailServer();
	johnMailClient = new MailClient(server, JOHN);
	paulMailClient = new MailClient(server, PAUL);
	assertTrue(server != null);
	assertTrue(johnMailClient != null);
	assertTrue(paulMailClient != null);
	assertEquals(JOHN,johnMailClient.getUser());
	assertEquals(PAUL,paulMailClient.getUser());
		true
		true
		true
		true
		true
			true
			true
			true
			true
			true
				✔

✔	//void sendMailItemTest()
	johnMailClient.sendMailItem(PAUL, MSG);
	MailItem item = server.getNextMailItem(PAUL);
	assertTrue(item != null);
	assertEquals(MSG,item.getMessage());
	assertEquals(JOHN,item.getFrom());
	assertEquals(PAUL,item.getTo());
		true
		true
		true
		true
			true
			true
			true
			true
				✔

✔	//getNextMailItemTest()
	johnMailClient.sendMailItem(PAUL, MSG);
	MailItem item = server.getNextMailItem(PAUL);
	//testValidItem(item);
	assertTrue(item != null);
	assertEquals(MSG,item.getMessage());
	assertEquals(JOHN,item.getFrom());
	assertEquals(PAUL,item.getTo());
	//No more emails
	item = server.getNextMailItem(PAUL);
	assertTrue(item == null);
		true
		true
		true
		true
		true
			true
			true
			true
			true
			true
				✔

✔	//void getNextMailItemTest
	server.post(new MailItem(JOHN, PAUL, MSG));
	//paulMailClient test
	MailItem item = paulMailClient.getNextMailItem();
	//testValidItem(item);
	assertTrue(item != null);
	assertEquals(MSG, item.getMessage());
	assertEquals(JOHN,item.getFrom());
	assertEquals(PAUL,item.getTo());
	item = paulMailClient.getNextMailItem();
	assertTrue(item == null);
		true
		true
		true
		true
		true
			true
			true
			true
			true
			true
				✔

	Test	Résultat attendu	Résultat obtenu	

✔	`//void doubleMessagesSentTest()`
	`johnMailClient.sendMailItem(PAUL, "1_"+ MSG);`
	`paulMailClient.sendMailItem(JOHN, "2_" + MSG);`
	`johnMailClient.sendMailItem(PAUL, "3_"+MSG);`
	`paulMailClient.sendMailItem(JOHN, "4_" + MSG);`
	`//John read first`
	`String itemToJohn = johnMailClient.nextMailItemToString();`
	`assertEquals("from='Paul', to='John', message='2_Salut!'",itemToJohn);`
	`//Paul read`
	`String itemToPaul = paulMailClient.nextMailItemToString();`
	`assertEquals("from='John', to='Paul', message='1_Salut!'", itemToPaul);`
	`//Paul read again`
	`itemToPaul = paulMailClient.nextMailItemToString();`
	`assertEquals("from='John', to='Paul', message='3_Salut!'", itemToPaul);`
	`//John read last`
	`itemToJohn = johnMailClient.nextMailItemToString();`
	`assertEquals("from='Paul', to='John', message='4_Salut!'", itemToJohn);`
		`true`
		`true`
		`true`
		`true`
			`true`
			`true`
			`true`
			`true`
				✔

Tous les tests ont été réussis ! ✔

▸ **Solution de l'auteur de la question (Java)**

Correct

Note pour cet envoi : 5,00/5,00.

Question 5

Partiellement correct

Note de 1,00 sur 6,00

On souhaite pouvoir ajouter **un sujet à certains emails, mais pas à tous.**

Vous n'avez pas le droit de modifier la classe *MailItem*, ni la classe *MailServer* et tous les précédents tests doivent continuer à fonctionner.

1. Voici la méthode que vous devez impléménter dans la classe **MailClient**

```
/**
 * Send the given message to the given recipient via the attached mail
 * server.
 *
 * @param to      The intended recipient.
 * @param subject      The subject
 * @param message The text of the message to be sent.
 * @return true if the item was sent; false otherwise.
 */
boolean sendMailItem(String to, String subject, String message) {
```

```
}
```

2. L'affichage d'un mail contenant un sujet doit afficher le sujet, par exemple :

from='Wilma', to='Fred', message='Did you remember to pick up the mammoth for supper?', subject='supper'

3. vous pouvez créer de nouvelles classes à votre convenance.

Copiez **tout le code de la classe MailClient et éventuellement des nouvelles classes et** collez-les dans la boîte de réponse .

RAPPEL : Ne modifiez aucune des autres classes (MailServer et MailItem) ; ne soumettez aucune de ces autres classes.

Vous pouvez récupérer ici des tests.

English Below

We want to add **a subject to some emails.**

You are not allowed to modify the MailItem class or the MailServer class, and all the previous tests must continue to work.

1. Here is the method you have to implement in the MailClient class

```
/**
 * Send the given message to the given recipient via the attached mail
 * server.
 *
 * @param to      The intended recipient.
 * @param subject      The subject
 * @param message The text of the message to be sent.
 * @return true if the item was sent; false otherwise.
 */
boolean sendMailItem(String to, String subject, String message) {
```

```
}
```

2. Displaying a mail containing a subject must display the subject, for example

from='Wilma', to='Fred', message='Did you remember to pick up the mammoth for supper?', subject='supper'

3. You can create new classes as you like.

Copy all the code from the ClientMail class and any new classes and paste it into the reply box.

REMINDER: Do not modify any of the other classes (MailServer and MailItem).

You can get some tests here.

Réponse : (régime de pénalités : 0 %)

```
    }

    /**
     * Return the next mail item (if any) for this user presents on the linked server
     * else return null;
     */
    @Override
    public MailItem getNextMailItem() {
        return server.getNextMailItem(user);
    }

    /**
     * Returns the next mail item (if any) for this user as a string
     */
    @Override
    public String nextMailItemToString() {
        MailItem item = getNextMailItem();
        if (item == null) {
            return "No new mail.";
        } else {
            return item.toString();
        }
    }

    /**
     * Send the given message to the given recipient via the attached mail
     * server.
     *
     * @param to      The intended recipient.
     * @param message The text of the message to be sent.
     * @return true if the item was sent; false otherwise.
     */
    @Override
    public boolean sendMailItem(String to, String message) {
        MailItem item = new MailItem(user, to, message);
        return server.post(item);
    }


    //It is better to work on the tests than on the hand!  These codes are only there to help understand.
    public static void main(String[] args) {
        // ---------------
        String JOHN = "John";
        String PAUL = "Paul";
        String MSG = "Yo! Ni hao! Tervist!";
        MailServer server = new MailServer();
        MailClient johnMailClient = new MailClient(server, JOHN);
        MailClient paulMailClient = new MailClient(server, PAUL);
        johnMailClient.sendMailItem(PAUL, MSG);
        johnMailClient.sendMailItem(PAUL, MSG);
        System.out.println(paulMailClient.nextMailItemToString());
        System.out.println(paulMailClient.nextMailItemToString());
```

	Test	Résultat attendu	Résultat obtenu	

✖	johnMailClient.sendMailItem(PAUL, SUBJECT, MSG);
	MailItem item = server.getNextMailItem(PAUL);
	assertTrue(item != null);
	assertEquals(MSG, item.getMessage());
	assertEquals(JOHN, item.getFrom());
	assertEquals(PAUL, item.getTo());
	assertTrue(item.toString().contains(SUBJECT));
		true
		true
		true
		true
		true
			true
			false
			true
			true
			true
				✖

✔	//les envois de messages "classiques" sont toujours possibles.
	johnMailClient.sendMailItem(PAUL, "1_"+ MSG);
	johnMailClient.sendMailItem(PAUL, "3_"+MSG);
	String itemToPaul = paulMailClient.nextMailItemToString();
	assertEquals("from='John', to='Paul', message='1_Salut!'", itemToPaul);
	itemToPaul = paulMailClient.nextMailItemToString();
	assertEquals("from='John', to='Paul', message='3_Salut!'", itemToPaul);
		true
		true
			true
			true
				✔

Montrer les différences

▾ **Solution de l'auteur de la question (Java)**

```
package test3.mailsystem;
/**
 * A class to model a simple email client. The client is run by a particular
 * user, and sends and retrieves mail via a particular server.
 *
 * @author David J. Barnes and Michael Kolling
 * @author MBF
 */
```

```
 */

public class MailClient implements MailClientInterface {
    //The server variable used for sending and receiving.
    final MailServer server;

    //The user variable running this client.
    private final String user;

    /**
     * Create a mail client run by user and dependent on the the given server.
     */
    MailClient(MailServer server, String user) {
        this.server = server;
        this.user = user;
    }

    /**
     * Return the next mail item (if any) for this user presents on the linked server
     * else return null;
     */
    @Override
    public MailItem getNextMailItem() {
        return server.getNextMailItem(user);
    }

    /**
     * Returns the next mail item (if any) for this user as a string
     */
    @Override
    public String nextMailItemToString() {
        MailItem item = getNextMailItem();
        if (item == null) {
            return ("No new mail.");
        } else {
            return (item.toString());
        }
    }

    /**
     * Send the given message to the given recipient via the attached mail
     * server.
     *
     * @param to      The intended recipient.
```

Partiellement correct

Note pour cet envoi : 1,00/6,00.

Question 6

Correct

Note de 7,00 sur 7,00

On souhaite pouvoir envoyer des **emails à des groupes.**

Dans le répertoire ci-joint, vous trouverez l'interface qui définit la notion de groupe.

1) Vous devez l'implémenter dans une classe **GroupImpl .**

2) Modifier la classe **MailClient** pour qu'elle implémente la méthode suivante :

```
/**
 * Send the given message to all the members of the group via the attached mail
 * server.
 *
 * @param to      The group of recipients.
 * @param message The text of the message to be sent.
 * @return true if the item was sent to all the recipient; false otherwise.
 */

public boolean sendMailItem(Group to, String message) {
    return true;
}
```

3) Copiez tout le code des classes **MailClient, GroupImpl et éventuellement de nouvelles classes** et collez-les dans la boîte de réponse.

Des Tests sont disponibles ici.

English Version

We want to be able to send emails to groups.

In the attached directory,you will find the interface that defines the notion of Group.

1) You must implement it in a **GroupImpl** class.

2) Modify the MailClient class so that it implements the following method :

```
/**
 * Send the given message to all the members of the group via the attached mail
 * server.
 *
 * @param to      The group of recipients.
 * @param subject      The subject
 * @param message The text of the message to be sent.
 * @return true if the item was sent to all the recipient; false otherwise.
 */

public boolean sendMailItem(Group to, String message) {
 return true;
}
```

3) Copy all the code from the **MailClient** class, **GroupImpl** and possibly new classes and paste it into the response box.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  // A CONSERVER
2
3  package test3.mailsystem;
4
5  import java.util.List;
6  import java.util.ArrayList;
7  import java.util.Map;
8  import java.util.Arrays;
9  import java.util.HashMap;
10
11 //--------
12
13 public class GroupImpl implements Group {
14     private final String name;
15     private final List<String> members;
```

```
    private final List<String> members;

    public GroupImpl(String name) {
        this.name = name;
        this.members = new ArrayList<>();
    }

    public List<String> members() {
        return members;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void addMember(String member) {
        members.add(member);
    }

    public String[] getMembers() {
        return members.toArray(new String[0]);
    }
}

public class MailClient implements MailClientInterface {
    //The server used for sending and receiving.
    private MailServer server;

    //The user  running this client.
    private String user;

    /**
     * Create a mail client run by user and dependent on the the given server.
     */
    MailClient(MailServer server, String user) {
        this.server = server;
```

	Test	Résultat attendu	Résultat obtenu	
✔	`group = new GroupImpl("Friends");` `assertEquals("Friends", group.getName());`	`true`	`true`	✔

✔	`group = new GroupImpl("Friends");`
	`group.addMember("Emma");`
	`String[] members = group.getMembers();`
	`assertEquals(1, members.length);`
		`true`	`true`	✔

✔	`String [] expected = new String[]{"Emma","Pierre","Yassine"};`
	`Arrays.sort(expected);`
	`String[] members = group.getMembers();`
	`Arrays.sort(members);`
	`assertEquals(3, members.length);`
	`assertTrue(Arrays.equals(expected,members));`
		`true`
		`true`
			`true`
			`true`
				✔

✔	`Map<String, MailClient> mailClients = new HashMap<>();`
	`String SUBJECT = "To say Hello";`
	`List<String> persons = new ArrayList<>(Arrays.asList("Emma","Pierre","Yassine","Paul",JOHN));`
	`server = new MailServer();`
	`for (String person : persons){`
	`  mailClients.put(person, new MailClient(server,person));`
	`}`
	
	`mailClients.get(JOHN).sendMailItem(group, MSG);`
	`for (String member : group.getMembers()){`
	`  MailItem item = mailClients.get(member).getNextMailItem();`
	`  assertEquals(JOHN,item.getFrom());`
	`  assertEquals(member,item.getTo());`
	`  assertEquals(MSG,item.getMessage());`
	`}`
		`true`
		`true`
		`true`
		`true`
		`true`
		`true`
		`true`
		`true`
		`true`
			`true`
			`true`
			`true`
			`true`
			`true`
			`true`
			`true`
			`true`
			`true`
				✔

✔	`String SUBJECT = "To say Hello";`
	`johnMailClient.sendMailItem(new GroupImpl("empty"), MSG);`
	`//"For an empty group, no mail are sent"`
	`assertEquals(0,server.howManyMailItems());`
		`true`	`true`	✔

Tous les tests ont été réussis ! ✔

▸ **Solution de l'auteur de la question (Java)**

Correct

Note pour cet envoi : 7,00/7,00.