Université Paris Cité

Rapport du Projet S5.08 : Qualité de développement

WU PATRICK
YE ERIC
GROUPE : 302

2023-2024

Sommaire

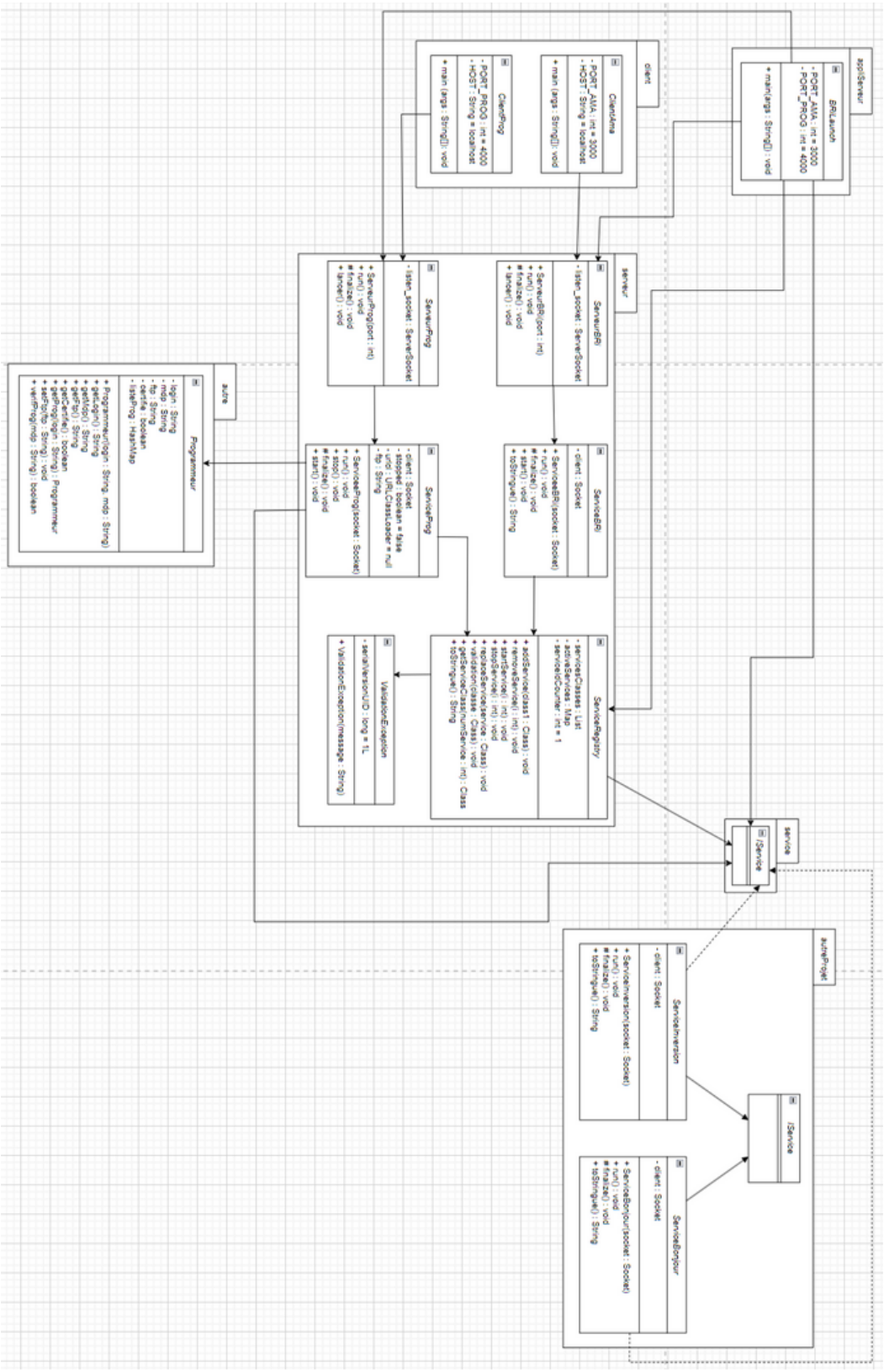
Introduction du projet

Le projet BRiLaunch vise à explorer la complexité des services dynamiques. Cette initiative se présente comme une interface d'échanges entre des programmeurs Java spécialisés dans le développement de services variés tels que la messagerie électronique, l'analyse syntaxique XML, la journalisation des activités, etc., et des non-programmeurs, désignés ici comme "amateurs," qui ont besoin de ces services.

L'objectif principal est de créer une plateforme dynamique, le serveur Java BRiLaunch, capable d'accueillir à la fois les programmeurs et les amateurs. Les programmeurs peuvent fournir, mettre à jour, démarrer, arrêter et désinstaller leurs services de manière distante, sans nécessiter le redémarrage du serveur BRiLaunch. Chaque programmeur dispose d'un serveur FTP dédié pour le stockage de ses services, et le déploiement est géré de manière transparente.

En conclusion, ce projet offre une opportunité d'approfondir les compétences en développement Java, en gestion de services dynamiques, et en conception de plateformes interactives, tout en répondant aux besoins des programmeurs et amateurs au sein de l'écosystème BRiLaunch.

Graphe de dépendance



Présentation du projet

On commence par lancer le serveur BRiLaunch.

Le serveur maintenant lancer, les programmeurs peuvent maintenant ajouter des services en se connectant via “ClientProg”. Il demandera le login, le mot de passe et si c’est un nouveau compte, le serveur FTP du programmeur.

```
Connecté au serveur localhost/127.0.0.1:4000
Bonjour, quel est votre login ?
Patrick
Quel est le mot de passe ?
Wu
Votre compte a été créé avec succès, quel est l'adresse de votre serveur FTP ?
ftp://localhost:2121/
```

Après cela, nous avons la liste des services qui sont en ligne ainsi que toutes les options disponibles au programmeur. Le programmeur choisit une option en tapant son numéro.

```
Activités présentes :

Que voulez vous faire ?
1. Fournir un nouveau service
2. Mettre à jour un service
3. Déclarer un changement d’adresse de son serveur ftp
4. Arrêter un service
5. Démarrer un service
6. Désinstaller un service
```

Commençons par ajouter le service “ServiceInversion” en tapant son package suivit de son nom. Dans le projet, nous n’avons pas réussi à avoir un package spécifique pour chaque programmeur, tous les services seront donc dans un package “service” et implémenteront l’interface “Service”.

```
Que voulez vous faire ?
1. Fournir un nouveau service
2. Mettre à jour un service
3. Déclarer un changement d’adresse de son serveur ftp
4. Arrêter un service
5. Démarrer un service
6. Désinstaller un service
1
Quel est le nom du service à installer ?
service.ServiceInversion
```

Maintenant pour utiliser ce service, nous pouvons lancer la class “ClientAma” qui affichera la liste des services disponibles.

```
ClientAma [Java Application] C:\Users\Patrick Wu\.p2\pool\plugins\org.eclipse.j
Connecté au serveur localhost/127.0.0.1:3000
Activités présentes :
1 Inversion de texte

Tapez le numéro de service désiré :
1
Tapez un texte à inverser
Bonjour, je m'appelle Patrick
kcirtaP elleppa'm ej ,ruojnoB
```

La mise à jour d’un service se passe comme l’ajout d’un service. Il suffit de taper le nom du package ainsi que le nom du service.

```
Que voulez vous faire ?
1. Fournir un nouveau service
2. Mettre à jour un service
3. Déclarer un changement d’adresse de son serveur ftp
4. Arrêter un service
5. Démarrer un service
6. Désinstaller un service
2
Quel est le nom du service que voulez vous mettre à jour ?
service.ServiceInversion
```

Bien sûr, lorsqu'un compte existe déjà, nous n'avons pas besoin d'entrer à nouveau son adresse ftp, en cas de faute, nous pouvons le changer directement en prenant la troisième option.

```
Que voulez vous faire ?
1. Fournir un nouveau service
2. Mettre à jour un service
3. Déclarer un changement d'adresse de son serveur ftp
4. Arrêter un service
5. Démarrer un service
6. Désinstaller un service
3
Voici votre adresse ftp actuelle :ftp://localhost:2121/. Quel est la nouvelle adresse ftp ?
ftp://localhost:2121/
```

L'arrêt d'un service se fait en fournissant le numéro du service via la liste des activités présentes. Pour démarrer un service, il doit être arrêté d'abord et une liste d'activité arrêtée est affichée. Il suffit de fournir le numéro du service arrêté pour le démarrer.
Pour désinstaller un service, cela se fait comme pour l'arrêt d'un service, la seule différence est qu'elle n'est plus démarrable et elle est supprimée de la liste des activités présentes.

Bilan du projet

Dans l'ensemble le projet s'est déroulé sans soucis, nous sommes un bon binôme car nous savons où nous allons, nous proposons des bonnes pistes et nous savons ce qu'il faut faire pour le projet. Cela est un point fort pour ce genre de projet, car on ne sait pas par où commencer parfois et on peut très vite se perdre.

1 Ce qui a été fait et réussi

Le minimum du cahier des charges a été quasiment complété avec deux services simples. Un service d'inversion de texte ainsi qu'un service qui dit bonjour. En matière d'options, nous avons fait le démarrage, l'arrêt ainsi que la désinstallation d'un service. Le service est totalement dynamique.

2 Ce qui n'a pas été fait ou pas réussi

Par rapport au minimum du cahier des charges, il manque le fait que chaque service soit dans un package qui porte le nom du programmeur. En effet pour que le projet marche, chaque service doit être dans un package "service" et doit implémenter l'interface "service". Hormis ce que j'ai cité dans ce qui a été fait et réussi, toutes les autres options n'ont pas été faites.

3 Ce qu'il pourrait être amélioré

La communication pourrait être améliorée. Nous avons tous les deux travaillé de notre côté. Malheureusement le niveau du groupe est très hétérogène. Nous avons pour cela du mal à nous coordonner, et cela peut amener à une discorde. Mais à part cela, nous nous aidons à accomplir les tâches demandées.

Conclusion

En conclusion, le projet BRiLaunch est une plateforme dynamique permettant aux programmeurs Java spécialisés et aux amateurs d'interagir de manière transparente. Malgré quelques lacunes, notamment l'absence de l'organisation des services dans des packages individuels, nous avons réalisé le minimum du cahier des charges avec succès en développant deux services simples et en implémentant des fonctionnalités telles que le démarrage, l'arrêt, et la désinstallation à distance.
La collaboration entre les membres du groupe a été solide, bien que la communication puisse être améliorée en raison de la disparité des niveaux de compétence.
Dans l'ensemble, le projet a offert une opportunité précieuse pour approfondir les compétences en développement Java, en gestion de services dynamiques, et en conception de plateformes interactives.

Code source

Package : appliServeur : classe BRiLaunch

```
package appliServeur;

import serveur.ServeurBRi;
import serveur.ServeurProg;

public class BRiLaunch {
 private final static int PORT_AMA = 3000;
 private final static int PORT_PROG = 4000;

 public static void main(String[] args) {

  System.out.println("Bonjour !");

  new Thread(new ServeurBRi(PORT_AMA)).start();
  new Thread(new ServeurProg(PORT_PROG)).start();

  //ftp://localhost:2121/
 }
}
```

Package : autre : classe Programmeur

```
package autre;

import java.util.HashMap;

public class Programmeur {
 private String login;
 private String mdp;
 private String ftp;
 private boolean certifie;
 private final static HashMap<String, Programmeur> listeProg = new HashMap<>();

 public Programmeur(String login, String mdp) {
  this.login = login;
  this.mdp = mdp;
  this.ftp = "";
  this.certifie = false;

  if(!listeProg.containsKey(login)) {
   listeProg.put(login, this);
  }
 }

 public String getLogin() {
  return this.login;
 }

 public String getMdp() {
  return this.mdp;
 }

 public String getFtp() {
  return this.ftp;
 }

 public boolean getCertifie() {
  return this.certifie;
 }

 public static Programmeur getProg(String login) {
  return listeProg.get(login);
 }

 public void setFtp(String ftp) {
  this.ftp = ftp;
  this.certifie = true;
 }

 public boolean verifProg(String mdp) {
  return (this.mdp.equals(mdp));
 }

}
```

Package : client : classe ClientAma

```
package client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientAma {

 private final static int PORT_AMA = 3000;
 private final static String HOST = "localhost";

 public static void main(String[] args) {
  Socket s = null;
  try {
   s = new Socket(HOST, PORT_AMA);

   BufferedReader sin = new BufferedReader (new InputStreamReader(s.getInputStream ( )));
   PrintWriter sout = new PrintWriter (s.getOutputStream ( ), true);
   BufferedReader clavier = new BufferedReader(new InputStreamReader(System.in));

   System.out.println("Connecté au serveur " + s.getInetAddress() + ":"+ s.getPort());

   String line;

   boolean continuer = true;

   while (continuer) {
      line = sin.readLine();
      if (line != null) {
         System.out.println(line.replaceAll("##", "\n"));
         sout.println(clavier.readLine());
      } else {
         continuer = false; // Si le serveur n'a rien envoyé, la boucle s'arrête
      }
   }

  }
  catch (IOException e) { System.err.println("Fin de la connexion"); }
  // Refermer dans tous les cas la socket
  try { if (s != null) s.close(); }
  catch (IOException e2) { ; }
 }
}
```

Package : client : classe ClientProg

```
package client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientProg {

 private final static int PORT_PROG = 4000;
 private final static String HOST = "localhost";

 public static void main(String[] args) {
  Socket s = null;
  try {
   s = new Socket(HOST, PORT_PROG);

   BufferedReader sin = new BufferedReader (new InputStreamReader(s.getInputStream ( )));
   PrintWriter sout = new PrintWriter (s.getOutputStream ( ), true);
   BufferedReader clavier = new BufferedReader(new InputStreamReader(System.in));

   System.out.println("Connecté au serveur " + s.getInetAddress() + ":"+ s.getPort());

   String line;
   boolean continuer = true;

   while (continuer) {
      line = sin.readLine();
      if (line != null) {
         System.out.println(line.replaceAll("##", "\n"));
         sout.println(clavier.readLine());
      } else {
         continuer = false;
      }
   }

  }
  catch (IOException e) { System.err.println("Fin de la connexion"); }
  // Refermer dans tous les cas la socket
  try { if (s != null) s.close(); }
  catch (IOException e2) { ; }
 }

}
```

Package : serveur : classe ServeurBRi

```
package serveur;


import java.io.*;
import java.net.*;


public class ServeurBRi implements Runnable {
 private ServerSocket listen_socket;

 // Cree un serveur TCP - objet de la classe ServerSocket
 public ServeurBRi(int port) {
  try {
   listen_socket = new ServerSocket(port);
  } catch (IOException e) {
   throw new RuntimeException(e);
  }
 }

 // Le serveur ecoute et accepte les connections.
 // pour chaque connection, il cree un ServiceInversion,
 // qui va la traiter.
 public void run() {
  try {
   while(true)
    new ServiceBRi(listen_socket.accept()).start();
  }
  catch (IOException e) {
   try {this.listen_socket.close();} catch (IOException e1) {}
   System.err.println("Pb sur le port d'écoute :"+e);
  }
 }

  // restituer les ressources --> finalize
 protected void finalize() throws Throwable {
  try {this.listen_socket.close();} catch (IOException e1) {}
 }

 // lancement du serveur
 public void lancer() {
  (new Thread(this)).start();
 }
}
```

Package : serveur : classe ServeurProg

```
package serveur;


import java.io.*;
import java.net.*;


public class ServeurProg implements Runnable {
 private ServerSocket listen_socket;

 // Cree un serveur TCP - objet de la classe ServerSocket
 public ServeurProg(int port) {
  try {
   listen_socket = new ServerSocket(port);
  } catch (IOException e) {
   throw new RuntimeException(e);
  }
 }

 // Le serveur ecoute et accepte les connections.
 // pour chaque connection, il cree un ServiceInversion,
 // qui va la traiter.
 public void run() {
  try {
   while(true)
    new ServiceProg(listen_socket.accept()).start();
  }
  catch (IOException e) {
   try {this.listen_socket.close();} catch (IOException e1) {}
   System.err.println("Pb sur le port d'écoute :"+e);
  }
 }

  // restituer les ressources --> finalize
 protected void finalize() throws Throwable {
  try {this.listen_socket.close();} catch (IOException e1) {}
 }

 // lancement du serveur
 public void lancer() {
  (new Thread(this)).start();
 }
}
```

Package : serveur : classe ServiceBRi

```
package serveur;


import java.io.*;
import java.lang.reflect.InvocationTargetException;
import java.net.*;

import service.Service;


class ServiceBRi implements Runnable {

 private Socket client;

 ServiceBRi(Socket socket) {
  client = socket;
 }

 public void run() {
  try {BufferedReader in = new BufferedReader (new
InputStreamReader(client.getInputStream ( )));
   PrintWriter out = new PrintWriter (client.getOutputStream ( ), true);
   out.println(ServiceRegistry.toStringue()+"##Tapez le numéro de service désiré :");
   int choix = Integer.parseInt(in.readLine());

   Class<? extends Service> classe = ServiceRegistry.getServiceClass(choix);
   Service service = classe.getConstructor(java.net.Socket.class).newInstance(this.client);
   service.run();
   }
  catch (IOException | InstantiationException | IllegalAccessException |
IllegalArgumentException | InvocationTargetException | NoSuchMethodException |
SecurityException e) {
   //Fin du service
  }

 try {client.close();} catch (IOException e2) {}
 }

 protected void finalize() throws Throwable {
  client.close();
 }

 // lancement du service
 public void start() {
  (new Thread(this)).start();
 }

}
```

Package : serveur : classe ServiceProg

```
package serveur;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.URL;
import java.net.URLClassLoader;

import autre.Programmeur;
import service.Service;

public class ServiceProg implements Runnable{

 private Socket client;
 URLClassLoader urlcl = null;
 String ftp;

 ServiceProg(Socket socket) {
  client = socket;
 }

 public void run() {
  try {BufferedReader in = new BufferedReader (new
InputStreamReader(client.getInputStream ( )));
   PrintWriter out = new PrintWriter (client.getOutputStream ( ), true);
   out.println("Bonjour, quel est votre login ?");
   String login = in.readLine();
   out.println("Quel est le mot de passe ?");
   String mdp = in.readLine();

   Programmeur utilisateur = Programmeur.getProg(login);

   if(utilisateur == null) {
    Programmeur newUtilisateur = new Programmeur(login, mdp);
    out.println("Votre compte a été créé avec succès, quel est l'adresse de votre serveur
FTP ?");
    ftp = in.readLine();
    newUtilisateur.setFtp(ftp);
   }
   else if (utilisateur.verifProg(mdp)){
    ftp = utilisateur.getFtp();
   }
   else {
    out.println("Mot de passe incorrect");
    client.close();
   }
```

```
out.println(ServiceRegistry.toStringue()+"##Que voulez vous faire ? ##1. Fournir un nouveau service ##2. Mettre à jour un service ##3. Déclarer un changement d'adresse de son serveur ftp ##4. Arrêter un service ##5. Démarrer un service ##6. Désinstaller un service");
  int choix = Integer.parseInt(in.readLine());

  String fileNameURL = ftp;
  switch(choix) {
   case 1:
    out.println("Quel est le nom du service à installer ?");
    try {
     String classeName = in.readLine();
     urlcl = URLClassLoader.newInstance(new URL[] {new URL(fileNameURL)});
     ServiceRegistry.addService(urlcl.loadClass(classeName).asSubclass(Service.class));
    } catch (Exception e) {
     System.out.println(e);
    }
    break;
   case 2:
    out.println("Quel est le nom du service que voulez vous mettre à jour ?");
    try {
     String classeName = in.readLine();
     urlcl = URLClassLoader.newInstance(new URL[] {new URL(fileNameURL)});
     ServiceRegistry.replaceService(urlcl.loadClass(classeName).asSubclass(Service.class));
    } catch (Exception e) {
     System.out.println(e);
    }
    break;
   case 3:
    out.println("Voici votre adresse ftp actuelle :" + ftp + ". Quel est la nouvelle adresse ftp ?");
    utilisateur.setFtp(in.readLine());
    break;
   case 4:
    out.println("Quel est le numéro du service que vous voulez arrêter ?");
    int stop = Integer.parseInt(in.readLine());
    ServiceRegistry.stopService(stop);
    break;
   case 5:
    out.println(ServiceRegistry.toStringueStop() + "Quel est le numéro du service que vous voulez démarrer ?");
    int start = Integer.parseInt(in.readLine());
    ServiceRegistry.startService(start);
    break;
   case 6:
    out.println("Quel est le numéro du service que vous voulez désinstaller ?");
    int desinstaller = Integer.parseInt(in.readLine());
    ServiceRegistry.removeService(desinstaller);
    break;
   default:
    break;
  }

  }
 catch (IOException | IllegalArgumentException | SecurityException e) {
  //Fin du service
 }

 try {client.close();} catch (IOException e2) {}
}

protected void finalize() throws Throwable {
  client.close();
}

// lancement du service
public void start() {
 (new Thread(this)).start();
}
}
```

Package : serveur : classe ServiceRegistry

```
package serveur;

import java.lang.reflect.Constructor;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.util.ArrayList;
import java.util.List;

import service.Service;

public class ServiceRegistry {

 private static List<Class<?extends Service>> servicesClasses;
 private static List<Class<?extends Service>> servicesStop;

 static {
  servicesClasses = new ArrayList<Class<? extends Service>>();
  servicesStop =new ArrayList<Class<? extends Service>>();
 }

 public static void addService(Class<? extends Service> class1) throws
ValidationException {
    validation(class1);

    boolean existe = false;
    for (Class<? extends Service> existingClass : servicesClasses) {
        if (existingClass.getName().equals(class1.getName())) {
            existe = true;
            break;
        }
    }

    for (Class<? extends Service> existingClass : servicesStop) {
        if (existingClass.getName().equals(class1.getName())) {
            existe = true;
            break;
        }
    }

    if (!existe) {
        servicesClasses.add(class1);
    } else {
        throw new ValidationException("La classe de service existe déjà");
    }
 }
```

```
public static void removeService(int i){
    servicesClasses.remove(i-1);
 }

 public static void startService(int i){
  Class<? extends Service> servicetmp = servicesStop.get(i-1);
    servicesStop.remove(i-1);
    servicesClasses.add(servicetmp);
 }

 public static void stopService(int i){
  Class<? extends Service> servicetmp = servicesClasses.get(i-1);
    servicesClasses.remove(i-1);
    servicesStop.add(servicetmp);
 }


 public static void replaceService(Class<? extends Service> service) throws
ValidationException{
  validation(service);

     for (Class<? extends Service> existingClass : servicesClasses) {
        if (existingClass.getName().equals(service.getName())) {
         servicesClasses.remove(existingClass);
    servicesClasses.add(service);
        }else {
    throw new ValidationException("La classe de service n'est pas dans la liste");
        }
     }
 }

 // une méthode de validation renvoie void et lève une exception si non validation
 // surtout pas de retour boolean !
 private static void validation(Class<? extends Service> classe) throws
ValidationException {
  // cette partie pourrait être déléguée à un objet spécialisé
  // le constructeur avec Socket
  Constructor<? extends Service> c = null;
  try {
   c = classe.getConstructor(java.net.Socket.class);
  } catch (NoSuchMethodException e) {
   // transformation du type de l'exception quand l'erreur est détectée par ce biais
   throw new ValidationException("Il faut un constructeur avec Socket");
  }
  int modifiers = c.getModifiers();
  if (!Modifier.isPublic(modifiers))
   throw new ValidationException("Le constructeur (Socket) doit être public");
  if (c.getExceptionTypes().length != 0)
   throw new ValidationException("Le constructeur (Socket) ne doit pas lever
d'exception");
  // etc... avec tous les tests nécessaires

 }
```

```
public static Class<? extends Service> getServiceClass(int numService) {
 return servicesClasses.get(numService - 1);
}

// liste les activités présentes
 public static String toStringue() {
  String result = "Activités présentes :##";
  int i = 1;
  // foreach n'est qu'un raccourci d'écriture
  // donc il faut prendre le verrou explicitement sur la collection
  synchronized (servicesClasses) {
   for (Class<? extends Service> s : servicesClasses) {
    try {
     Method toStringue = s.getMethod("toStringue");
     String string = (String) toStringue.invoke(s);
     result = result + i + " " + string+"##";
     i++;
    } catch (Exception e) {
     e.printStackTrace(); // normalement déjà testé par validation()
    }
   }
  }
  return result;
 }

 public static String toStringueStop() {
  String result = "Activités arrêtées présentes :##";
  int i = 1;
  synchronized (servicesClasses) {
   for (Class<? extends Service> s : servicesStop) {
    try {
     Method toStringue = s.getMethod("toStringue");
     String string = (String) toStringue.invoke(s);
     result = result + i + " " + string+"##";
     i++;
    } catch (Exception e) {
     e.printStackTrace();
    }
   }
  }
  return result;
 }

}
```

Package : serveur : classe ValidationException

```
package serveur;

public class ValidationException extends Exception {
 private static final long serialVersionUID = 1L;

 public ValidationException(String message) {
  super(message);
 }

}
```

Package : service : classe interface Service

```
package service;

public interface Service extends Runnable {
}
```

Code source des services

Package : service : classe ServiceBonjour

```
package service;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ServiceBonjour implements Service{

 private final Socket client;

 public ServiceBonjour(Socket socket) {
  client = socket;
 }
 @Override
 public void run() {
  try {BufferedReader in = new BufferedReader (new InputStreamReader(client.getInputStream ( )));
   PrintWriter out = new PrintWriter (client.getOutputStream ( ), true);

   out.println("Bonjour, comment vous appelez vous ?");

   String line = in.readLine();

   out.println("Bonjour " + line + " !");

   client.close();
  }
  catch (IOException e) {

  }
 }

 protected void finalize() throws Throwable {
   client.close();
 }

 public static String toStringue() {
  return "Service de bonjour";
 }

}
```

Package : service : classe ServiceInversion

```
package service;

import java.io.*;
import java.net.*;


public class ServiceInversion implements Service {

 private final Socket client;

 public ServiceInversion(Socket socket) {
  client = socket;
 }
@Override
 public void run() {
  try {BufferedReader in = new BufferedReader (new
InputStreamReader(client.getInputStream ( )));
   PrintWriter out = new PrintWriter (client.getOutputStream ( ), true);

   out.println("Tapez un texte à inverser");

   String line = in.readLine();

   String invLine = new String (new StringBuffer(line).reverse());

   out.println(invLine);

   client.close();
  }
  catch (IOException e) {
   //Fin du service d'inversion
  }
 }

 protected void finalize() throws Throwable {
  client.close();
 }

 public static String toStringue() {
  return "Inversion de texte";
 }
}
```