**Préparation TP3 GL3**

Nom : AMIMER

Prénom : Abderrahmane Mohamed Elmahdi

Matricule : 202031049121

Group tp3

**Exercice 1:**

1)- le test de méthode tel que le mot de passe doit être entre 8 et 12 caractères :

- le test :

@Test

    void testIsValidated() {

        PasswordValidator p = new PasswordValidator() ;

        assertAll(

                ()->assertTrue("la methode isValidated a un erreur de cas retourne vrai",p.isValidated("abcdefghi")),

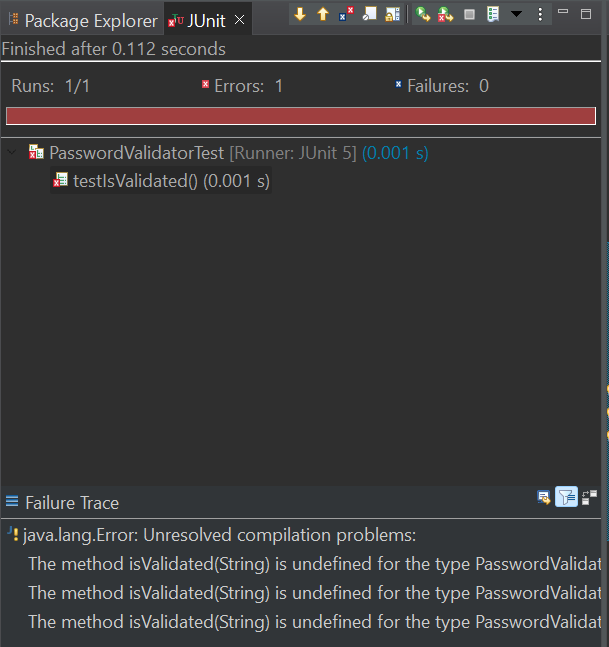
                ()->assertFalse("la methode isValidated a un erreur de cas password a moin de 8 caracteres",p.isValidated("abc")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a plus de 12 caracteres",p.isValidated("abcdefghijklmnopqr"))

              );

    }

Ce test est écrit avant la création de la méthode isValidated donc il va échoue :



- création de la méthode :

public boolean isValidated(String password) {

    int length = password.length();

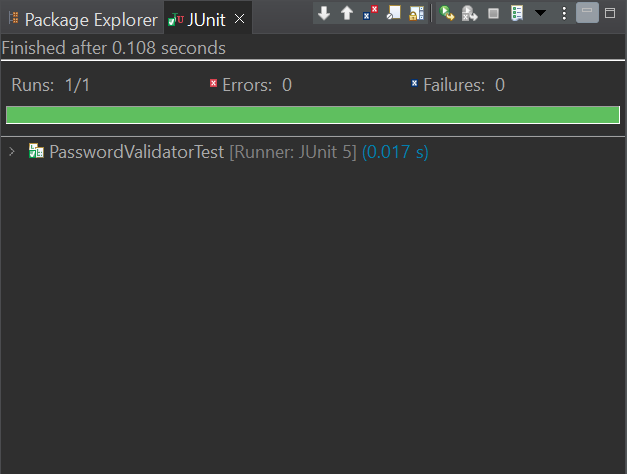
       if(length>=8 && length<=12) {

           return true;

       }else return false;

    }

- l’exécution de test avec le remaniement :



2)- le test de méthode tel que le mot de passe doit avoir un chiffre minimum :

- le test :

@Test

    void testIsValidated() {

        PasswordValidator p = new PasswordValidator() ;

        assertAll(

                ()->assertTrue("la methode isValidated a un erreur de cas retourne vrai",p.isValidated("abcdef3gh")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a moin de 8 caracteres",p.isValidated("abc")),

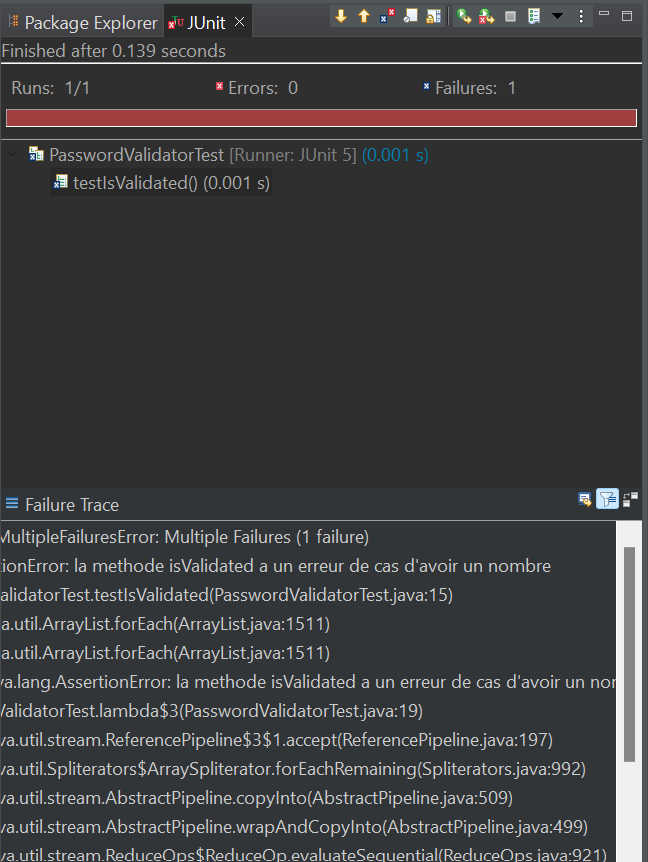
                ()->assertFalse("la methode isValidated a un erreur de cas password a plus de 12 caracteres",p.isValidated("abcdefghijklmnopqr")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un nombre ",p.isValidated("adefghibc"))

              );

    }

Ce test sera échoué dans la 4éme assertion :



- le code :

public boolean isValidated(String password) {

    int length = password.length();

       if(length>=8 && length<=12) {

           for (char c : password.toCharArray()) {

               if (Character.isDigit(c)) {

                   return true;

               }

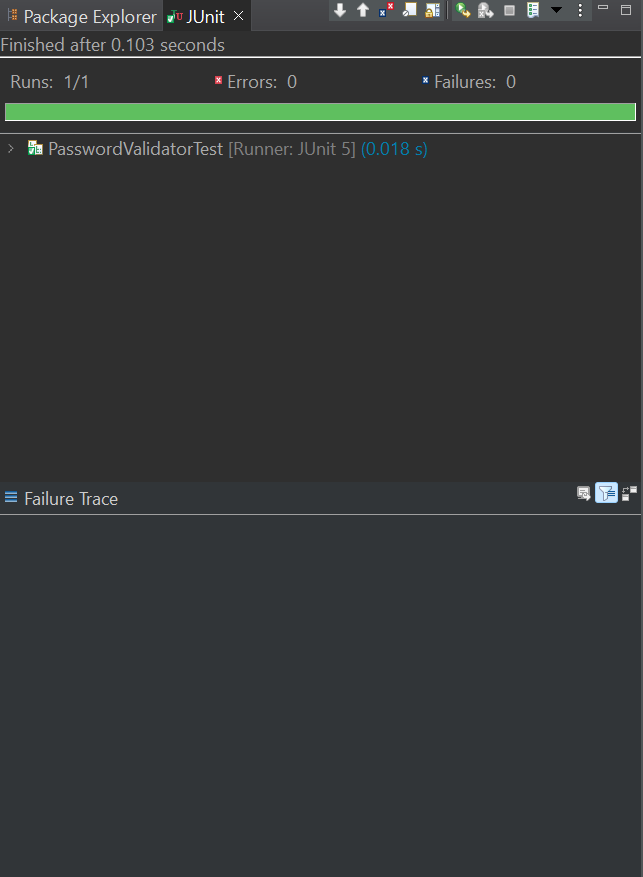
           }

           return false;

       }else return false;

    }

Et alors l’exécution de test :



3)- le test de méthode tel que le mot de passe doit avoir un caractère en minuscule minimum :

- le test :

@Test

    void testIsValidated() {

        PasswordValidator p = new PasswordValidator() ;

        assertAll(

                ()->assertTrue("la methode isValidated a un erreur de cas retourne vrai",p.isValidated("abcdef3gh")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a moin de 8 caracteres",p.isValidated("abc")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a plus de 12 caracteres",p.isValidated("abcdefghijklmnopqr")),

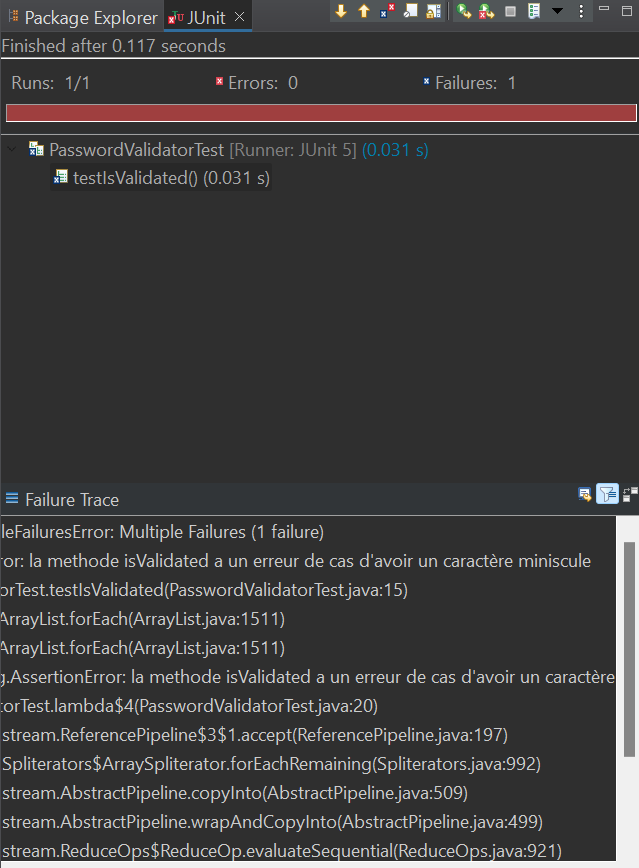
                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un nombre ",p.isValidated("adefghibc")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un caractère miniscule ",p.isValidated("AZERTYUI2"))

              );

    }

La dernière assertion sera échouée :



- la correction de code :

public boolean isValidated(String password) {

    int length = password.length();

    boolean hasNumber = false;

    boolean hasMiniscule = false;

       if(length>=8 && length<=12) {

           for (char c : password.toCharArray()) {

               if (Character.isDigit(c)) {

                   hasNumber = true;

               }else if(Character.isLowerCase(c)){

                   hasMiniscule = true;

               }

           }

           if (hasNumber && hasMiniscule) {

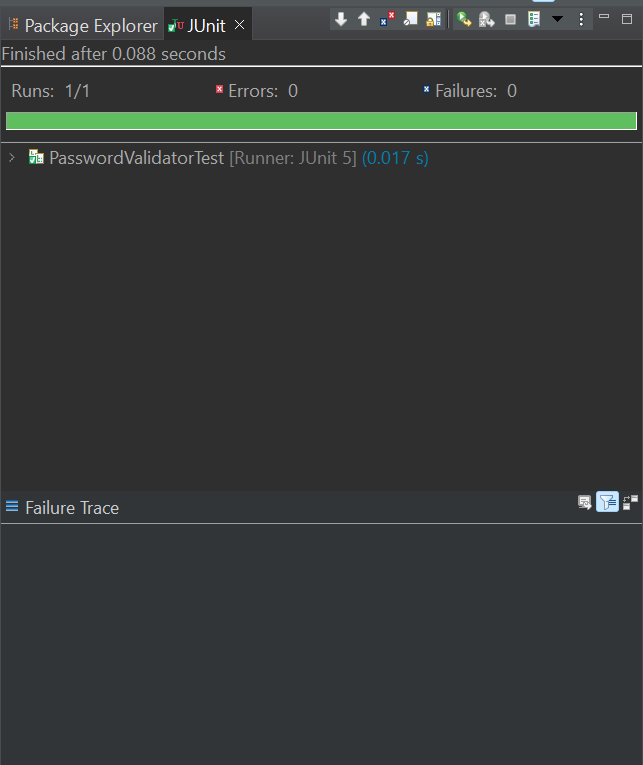
               return true;

           }else return false;

       }else return false;

    }

Et alors l’exécution de test :



4)- le test de méthode tel que le mot de passe doit avoir un caractère en majuscule minimum :

- le test :

@Test

    void testIsValidated() {

        PasswordValidator p = new PasswordValidator() ;

        assertAll(

                ()->assertTrue("la methode isValidated a un erreur de cas retourne vrai",p.isValidated("Abcdef3gh")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a moin de 8 caracteres",p.isValidated("Abc")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a plus de 12 caracteres",p.isValidated("Abcdefghijklmnopqr")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un nombre ",p.isValidated("Adefghibc")),

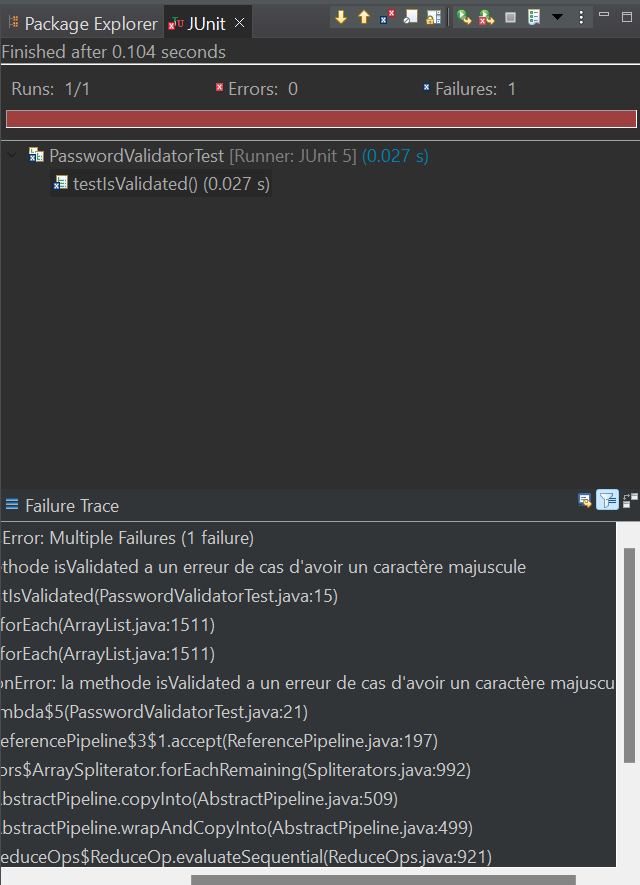
                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un caractère miniscule ",p.isValidated("AZERTYUI2")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un caractère majuscule ",p.isValidated("abcdefgh2"))

              );

    }

L’échoue de test :



- la correction de code :

public boolean isValidated(String password) {

    int length = password.length();

    boolean hasNumber = false;

    boolean hasMiniscule = false;

    boolean hasMajuscule = false;

       if(length>=8 && length<=12) {

           for (char c : password.toCharArray()) {

               if (Character.isDigit(c)) {

                   hasNumber = true;

               }else if(Character.isLowerCase(c)){

                   hasMiniscule = true;

               }else if(Character.isUpperCase(c)) {

                   hasMajuscule=true;

               }

           }

           if (hasNumber && hasMiniscule && hasMajuscule) {

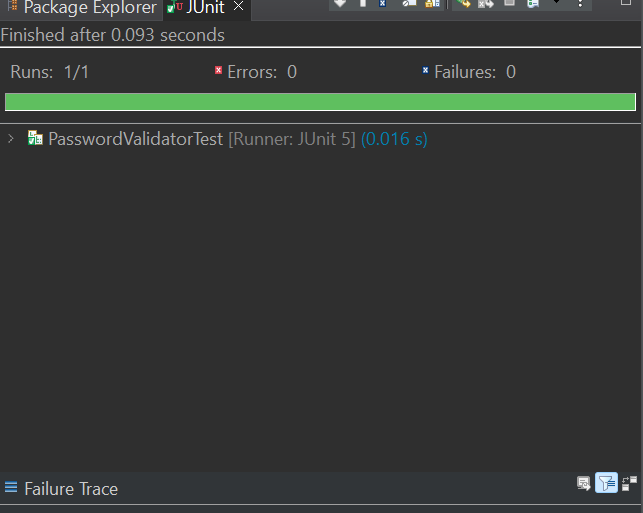
               return true;

           }else return false;

       }else return false;

    }

- l’exécution de test :



5)- le test de méthode tel que le mot de passe doit avoir un caractère spécial minimum :

- le test :

@Test

    void testIsValidated() {

        PasswordValidator p = new PasswordValidator() ;

        assertAll(

                ()->assertTrue("la methode isValidated a un erreur de cas retourne vrai",p.isValidated("Abcd@ef3gh")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a moin de 8 caracteres",p.isValidated("Abc(")),

                ()->assertFalse("la methode isValidated a un erreur de cas password a plus de 12 caracteres",p.isValidated("-Abcdefghijklmnopqr")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un nombre ",p.isValidated("Adefg&hibc")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un caractère miniscule ",p.isValidated("AZE@RTYUI2")),

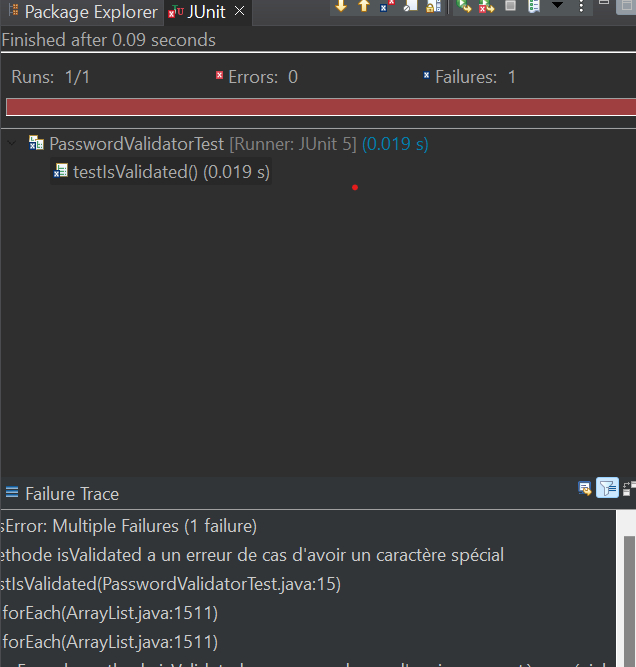
                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un caractère majuscule ",p.isValidated("a\_bcdefgh2")),

                ()->assertFalse("la methode isValidated a un erreur de cas d'avoir un caractère spécial ",p.isValidated("Abcdefgh2"))

              );

    }

- L’échoue dans la dernière assertion :



- la correction de code :

public boolean isValidated(String password) {

    int length = password.length();

    boolean hasNumber = false;

    boolean hasMiniscule = false;

    boolean hasMajuscule = false;

    boolean hasSpecialChar = false;

       if(length>=8 && length<=12) {

           for (char c : password.toCharArray()) {

               if (Character.isDigit(c)) {

                   hasNumber = true;

               }else if(Character.isLowerCase(c)){

                   hasMiniscule = true;

               }else if(Character.isUpperCase(c)) {

                   hasMajuscule=true;

               }else if(!Character.isLetterOrDigit(c)) {

                   hasSpecialChar = true;

               }

           }

           if (hasNumber && hasMiniscule && hasMajuscule && hasSpecialChar) {

               return true;

           }else return false;

       }else return false;

    }

- l’exécution du test :

