EXERCICE 1 : RELECTURE ET TEST STRUCTUREL

Question 1. Que fait la classe C ?

Question 2. Cette classe est-elle correcte ? Si non, où se trouve l'erreur ?

Question 3. Donner le graphe de flot de contrôle de la méthode x (vous pourrez renommer les variables).

Question 4. Donner des cas de test pour couvrir les 1-chemins.

```
public class C {

	private int[] l;
	private final static int s = 10;

	public C() {
		m();
		x(l.length);
	}

	private void m() {
		l = new int[s];
		Random r = new Random(System.currentTimeMillis());
		for(int i = 0; i < l.length; i++) {
			l[i] = Math.abs(r.nextInt() % s);
		}
	}

	private void n(int i, int k) {
		int t = l[i];
		l[i] = l[k];
	}

	public void x(int n) {
		System.out.println(n);
		boolean a = true;
		while(a) {
			a = false;
			for(int i = 1; i < n; i++) {
				if (l[i] < l[i - 1]) {
					n(i, i - 1);
					a = true;
				}}
			n = n -1;
		}
	}
}
```

EXERCICE 2 : JUNIT

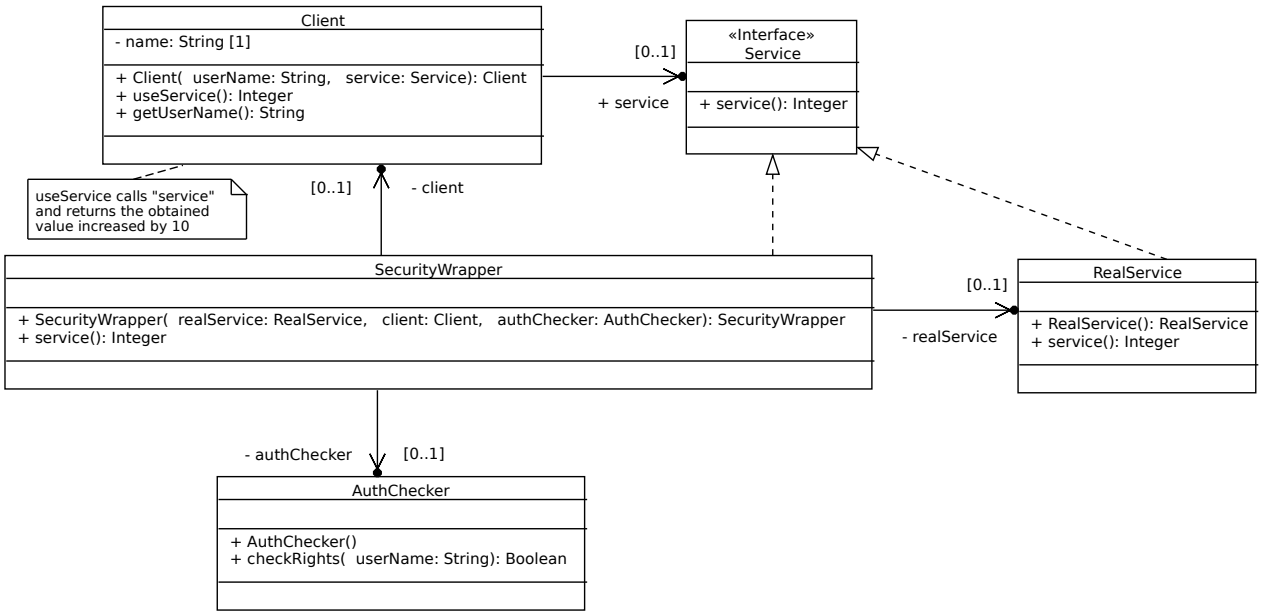
```
package test;

public class List {

    private int[] tab = new int[10];
    private int current = -1;

    public void add(int i) {
        current++;
        tab[current]=i;
    }

    public int length() {
        return current+1;
    }

    public int current() {
        return tab[current];
    }

    public int get(int i) {
        return tab[i];
    }

}
```

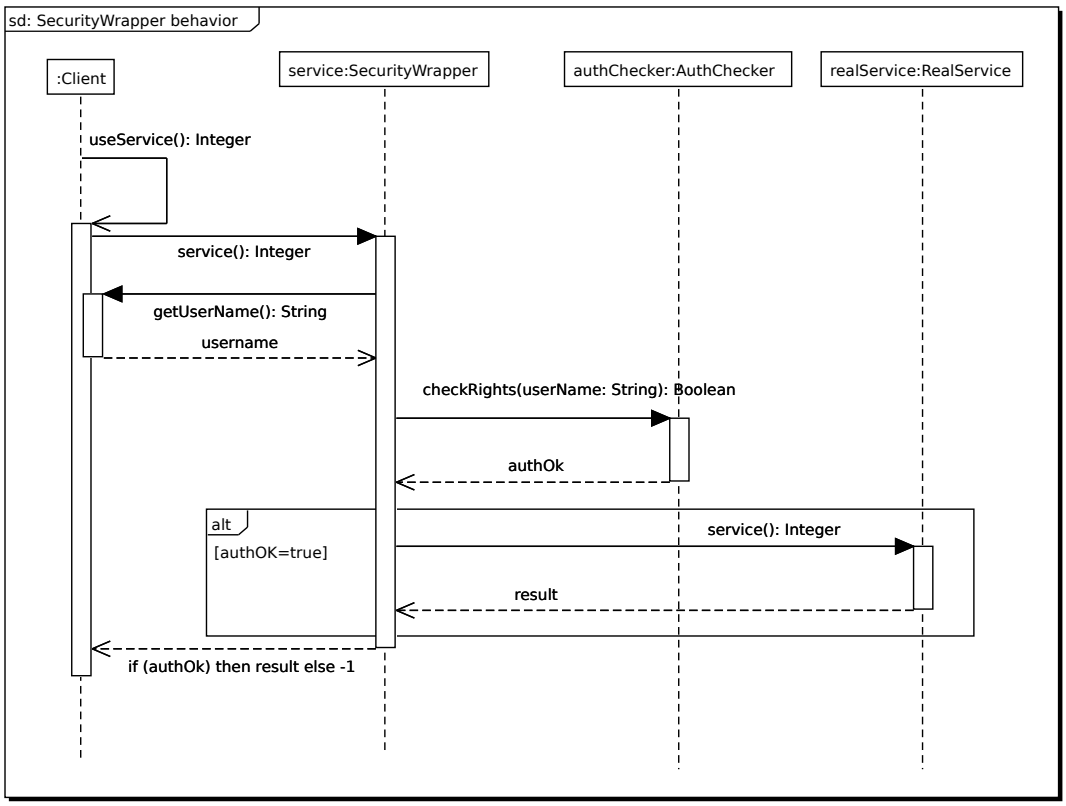
Question 5. Ecrire 4 cas de test JUnit pour la classe **`List`**.

```java
public class Stream {
    BufferedReader reader;

    public Stream(String fileNameData) throws
FileNotFoundException {
        this.reader = new BufferedReader(new
FileReader(fileNameData));
    }

    public void readData()  {
        try {
            String line = reader.readLine();
            while (line != null) {
                System.out.println(parseLine(line));
                line = reader.readLine();
            }
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    protected double parseLine(String line) {
        String[] split = line.split(";");
        return Double.parseDouble(split[0]) /
Double.parseDouble(split[1]);
    }

}
```

Question 6. Ecrire 2 cas de test JUnit pour la méthode `readData`, un cas de test qui ne passe que dans le try et un qui passe dans le catch.

EXERCICE 3 : INTEGRATION





Question 7. Tester le comportement de `service()`, du point de vue **`Client`**

EXERCICE 4 : TEST D'UN SYSTEME DE REGULATION DE VITESSE

Le système de régulation de vitesse d'un véhicule de la marque Citreugeot™ comporte quatre entrées. Le régulateur de vitesse agit sur l'accélérateur et utilise le frein moteur pour ralentir. Il ne contrôle ni le changement de vitesse, ni le système de frein.

- La première entrée est la vitesse courante (current speed). En dessous de 70km/h, le système ne peut pas être actif, au delà de 180 km/h non plus.
- La deuxième entrée est une action de déverrouillage/verrouillage du régulateur (block/unblock). Cette action est déclenchée soit par l'usage du pédalier, soit par le joystick, soit avec le déclenchement des air-bags, une panne moteur ou une panne frein.
- La troisième entrée est la commande « joystick » qui permet de faire varier par une pression d'un doigt la vitesse de ±1 ou ±5 km/h. Si aucune pression n'est exercée, la commande renvoie 0.
- La quatrième entrée est un indicateur de la capacité du frein moteur à ralentir la voiture. Cet indicateur sert à calculer si la vitesse demandée en cas de ralentissement peut être atteinte sans le frein. Il s'agit d'un entier variant entre 0 et 3 (0= pas besoin de freiner, 1= freinage moteur efficace, 2= freinage moteur lent, 3= freinage insuffisant).

Le système actionne l'accélérateur et calcule le temps probable de freinage. La sortie, outre les commandes vers le moteur que l'on ne détaille pas ici, est un afficheur de la vitesse demandée, qui peut être dans 4 modes :

- clignotant rapidement, la vitesse ne peut être atteinte (trop basse, trop haute, ou frein moteur insuffisant),
- clignotant lentement, la vitesse demandée peut être atteinte, mais cela se fera dans un délai supérieur à 5 secondes,
- pas de clignotement (la vitesse peut être atteinte en moins de 5 secondes).
- Un point rouge indique le moment où la vitesse demandée est effectivement atteinte.

Question 8. Faites une analyse partitionnelle (category-partition) sur les domaines de chaque variable d'entrée du système de régulation de vitesse. Pour les variables définis sur des domaines continus, on considèrera aussi, en les indiquant séparément, les valeurs aux limites (limit-testing).

Question 9. Un cas de test est un couple (valeurs d'entrée, résultat attendu). Proposer quatre cas de test nominaux permettant d'observer chacun des résultats attendus possibles. Est-ce que cela vous paraît suffisant ?

Question 10. On souhaite tester tous les cas possibles (exhaustive testing) à partir de l'analyse partitionnelle. Combien de cas de test sont-ils nécessaires sans tenir compte des valeurs aux limites ? Combien de cas de test sont-ils nécessaires avec les valeurs aux limites ?

Question 11. Appliquer le pair-wise testing sans tenir compte des valeurs aux limites.

Question 12. Sans calculer la solution explicitement, combien de cas de test seraient nécessaires en appliquant le pair-wise testing pour tester aussi les cas limites ?

Question 13. Dans ce type de test, un appel au régulateur correspond à un test. La dimension temporelle n'est pas prise en compte. Redéfinir la notion de cas de test. Donner un exemple. Une des entrées est inutile dans cette manière de tester et n'avait été introduite que pour les besoins du test. Laquelle ?

PAIR-WISE TESTING

Le pair-wise testing est une technique intermédiaire entre le test minimal et le test exhaustif sur le domaine d'entrée d'un programme. Soit un système S, prenant p paramètres d'entrée Pi. |P1| × |P2| × …. |Pp| => domaine d'entrée

Le pair-wise testing garantit que

∀ i et j, soit pi (resp. pj) une valeur du domaine Pi (resp. Pj), chaque paire de valeurs possibles (pi, pj) apparaît au moins une fois dans les tests.

Exemple : Soit une fonction f(a,b, c) telle que : a prend ses valeurs dans l'ensemble {1, 2, 3}, b={1, 2, 3} et c= {1, 2, 3} . Une solution du pair-wise testing est :

a	b	c
1	1	1
1	2	2
1	3	3
2	1	2
2	2	3
2	3	1
3	1	3
3	2	1
3	3	2

EXERCICE 5 : MUTATION

```
pow(x, y:integer) : float
  local i, p : integer
  i := 0;                          {1}
  Result := 1;                     {2}
  if y<0 then p := -y;             {a,3}
  else p := y;                     {4}
  while i<p do                     {b}
    Result := Result * x;          {5}
    i := i + 1;                    {6}
  done
  if y<0 then                      {c}
    Result := 1/Result;            {7}
end
```

Les numéros désignent des instructions et les lettres les prédicats.

On note P pour *Pass* lorsque le test ne détecte pas la faute et F pour *Fail* si il la détecte.

Si nécessaire, on interprétera les instructions sur les nombres rationnels et pas sur les flottants (on ne demande pas ici de calcul exact).

On souhaite appliquer la technique des mutants de la manière suivante : à partir du programme initial, on produit un mutant en effectuant *une (et une seule)* des modifications suivantes :

O1 : l'opérateur < est remplacé par <=

O2 : l'opérateur binaire / est remplacé par *

O3 : l'opérateur unaire - est remplacé par +

O4 : substitution de la variable y par x dans une instruction

On notera Oi(n°) les mutants, i étant le numéro de l'opérateur et n° désignant la ligne concernée par la mutation (lettre pour un prédicat).

Question 14. Construire le graphe de contrôle du programme. Produire un nombre minimum de données de test couvrant :

- tous les nœuds du graphe
- tous les arcs,
- tous les 0-chemins,
- tous les 1-chemins.

Question 15. Combien de mutants obtient-on ainsi ? Les lister.

Question 16. Y a-t-il des mutants équivalents ? Détailler les mutants tués par chaque donnée de test ci-dessous (il n'est pas nécessaire de calculer exactement le résultat de l'exécution de chaque mutant). Quel score de mutation obtient-on ?

DT1 = (x= 2, y = 4), DT2 = (x= -2 , y= 0), et DT3 = (x= -3, y= -3)

Combien de données supplémentaires faut-il pour atteindre un score de 100%.