Antoine Gicquel INFO1 A2
TP2 Rendu IHM 2105

A. Exceptions : première expérience

1) Le code de Tableau :

```
public class Tableau
{
	private int longueur;
	private int tab[];

	public Tableau(int longueur)
	{
		if(longueur >= 0)
		{
			this.longueur = longueur;
			this.tab = new int [longueur];
		}
		else
		{
		}
	}

	public int getTab(int i)
	{
		int ret = -1;
		//if(i >= 0 && i < this.longueur)
			ret = this.tab[i];
		return ret;
	}

	public void setTab(int i, int value)
	{
		if(i >= 0 && i < this.longueur)
			this.tab[i] = value;
	}
}
```

2) Les différentes erreurs :

- Une case d'indice négatif
  Lance une exception a l'exécution : « `java.lang.ArrayIndexOutOfBoundsException` »
- Une case d'indice supérieur à la taille du tableau :
  Lance une exception a l'exécution : « `java.lang.ArrayIndexOutOfBoundsException` »
- Une case sans avoir initialisé le tableau :
  Ne passe pas à la compilation : « `variable tab might not have been initialized` »

```
Tableau t = new Tableau(10);

// case indice negatif
t.getTab(-5);

// case indice supérieur à la taille du tableau
t.getTab(15);

// sans avoir initialisé le tableau
int tab[];
int r = tab[5];
```

3) Dans le cas des exceptions non traités, le programme est stoppé et affiche d'où vient le problème en montrer l'appel successif des méthodes ainsi que des paramètres. Par exemple :

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -5
       at Tableau.getTab(Tableau.java:22)
       at TestTableau.main(TestTableau.java:8)


Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 15
       at Tableau.getTab(Tableau.java:22)
       at TestTableau.main(TestTableau.java:11)
```

B. Instructions try/catch et throws

1) Le code du constructeur sans try/catch :

```
Scanner scanner = new Scanner(new File(fichierNom));

int taille = scanner.nextInt();
this.tab = new int[taille];
int i = 0;

while(scanner.hasNextInt() && i < taille)
{
     this.tab[i] = scanner.nextInt();
     i++;
}
```

2) Ne pas gérer les exceptions IOException provoque une erreur de compilation comme ci dissous car c'est indispensable parce que les risques d'innacibilité d'un fichier sont trop conséquentes.

```
../src/Tableau.java:24: error: unreported exception
FileNotFoundException; must be caught or declared to be thrown
                       Scanner scanner = new Scanner(new
File(fichierNom));
```

3) Dans la javadoc, les méthodes susceptibles de lancer une erreur ont une partie `Throws` pour chaque exception.

4) Voici le code de test : `Tableau t = new Tableau("tab");` alors qu'il n'existe pas de fichier tab. On obtient l'erreur :

```
java TestTableau
tab (Aucun fichier ou dossier de ce type)
```

Le code des catchs :

```
catch(FileNotFoundException e)
{
	System.out.println(e.getMessage());
	System.exit(-1);
}
catch(NegativeArraySizeException e)
{
	System.out.println(e.getMessage());
	System.exit(-1);
}
```

5) et 6) Le constructeur de Tableau :

```
public Tableau(String fichierNom) throws IOException
{
	Scanner scanner = new Scanner(new File(fichierNom));

	int taille = scanner.nextInt();
	this.tab = new int[taille];
	int i = 0;
	while(scanner.hasNextInt() && i < taille)
	{
		this.tab[i] = scanner.nextInt();
		i++;
	}
}
```

Le code de test :

```
try
{
	Tableau t = new Tableau("tab");
}
catch(IOException e)
{
	System.out.println(e.getMessage());
}
```

7) Les RuntimeException sont gérées nativement par java et donc le programmeur n'a pas l'obligation de les gérer, alors que les IOExceptions ont besoins d'être gérées.

C. Génération d'exceptions par le programmeur

1) Voici le code de l'exception :

```
public class ErrTableau extends Exception
{
	public ErrTableau(int err)
	{
		super("Taille de tableau invalide : " + err);
		printStackTrace();
```

```
	}
}
```

L’affichage en cas d’erreur :

```
ErrTableau: Taille de tableau invalide : -5
	at Tableau.<init>(Tableau.java:29)
	at TestTableau.main(TestTableau.java:29)
Taille de tableau invalide : -5
```

2) et 3) Modification de ErrTableau :

```
public class ErrTableau extends Exception
{
	public ErrTableau(int err)
	{
		super("Taille de tableau invalide : " + err);
		printStackTrace();
	}

	public ErrTableau(String err)
	{
		super(err);
		printStackTrace();
	}
}
```

Le code du constructeur de Tableau :

```
public Tableau(String fichierNom) throws ErrTableau, FileNotFoundException
{
	try
	{
		Scanner scanner = new Scanner(new File(fichierNom));

		int taille = scanner.nextInt();
		if(taille < 0)
			throw new ErrTableau(taille);

		this.tab = new int[taille];
		int i = 0;

		while(scanner.hasNextInt() && i < taille)
		{
			this.tab[i] = scanner.nextInt();
			i++;
		}
	}
	catch(FileNotFoundException e)
	{
		throw new ErrTableau("Probleme d'ouverture du fichier");
	}
}
```

Affichage de l’erreur d’exécution :

```
ErrTableau: Probleme d'ouverture du fichier
	at Tableau.<init>(Tableau.java:43)
	at TestTableau.main(TestTableau.java:29)
Probleme d'ouverture du fichier
```

D. Pour finir

1) Modification du constructeur :

```
public Tableau(String fichierNom) throws ErrTableau, FileNotFoundException
{
	Scanner scanner = null;
	try
	{
		scanner = new Scanner(new File(fichierNom));
		int taille = scanner.nextInt();
		if(taille < 0)
			throw new ErrTableau(taille);

		this.tab = new int[taille];
		int i = 0;

		while(scanner.hasNextInt() && i < taille)
		{
			this.tab[i] = scanner.nextInt();
			i++;
		}
	}
	catch(FileNotFoundException e)
	{
		throw new ErrTableau("Probleme d'ouverture du fichier");
	}
	finally
	{
		scanner.close();
	}
}
```

ANNEXE

Tableau.java :

```
import java.io.*;
import java.util.Scanner;

public class Tableau
{
	private int longueur;
	private int tab[];

	public Tableau(int longueur) throws ErrTableau
	{
		if(longueur >= 0)
		{
			this.longueur = longueur;
			this.tab = new int [longueur];
		}
		else
		{
			throw new ErrTableau(longueur);
		}
	}

	public Tableau(String fichierNom) throws ErrTableau,
FileNotFoundException
	{
		Scanner scanner = null;
		try
		{
			scanner = new Scanner(new File(fichierNom));

			int taille = scanner.nextInt();
			if(taille < 0)
				throw new ErrTableau(taille);

			this.tab = new int[taille];
			int i = 0;

			while(scanner.hasNextInt() && i < taille)
			{
				this.tab[i] = scanner.nextInt();
				i++;
			}
		}
		catch(FileNotFoundException e)
		{
			throw new ErrTableau("Probleme d'ouverture du fichier");
		}
		finally
		{
			scanner.close();
		}
	}

	/*
	public Tableau(String fichierNom) throws IOException
```

```
	{
		Scanner scanner = new Scanner(new File(fichierNom));

		int taille = scanner.nextInt();
		this.tab = new int[taille];
		int i = 0;
		while(scanner.hasNextInt() && i < taille)
		{
			this.tab[i] = scanner.nextInt();
			i++;
		}
	}
	*/

	public int getLongueur()
	{
		return this.longueur;
	}

	public int getTab(int i)
	{
		int ret = -1;
		//if(i >= 0 && i < this.longueur)
			ret = this.tab[i];
		return ret;
	}

	public void setTab(int i, int value)
	{
		if(i >= 0 && i < this.longueur)
			this.tab[i] = value;
	}
}
```

TableauTest.java :

```
import java.io.*;

public class TestTableau
{
	public static void main(String[] args)
	{
		Tableau t = new Tableau(10);

		// case indice negatif
		t.getTab(-5);

		// case indice supérieur à la taille du tableau
		t.getTab(15);

		// sans avoir initialisé le tableau
		int tab[];
		int r = tab[5];

		---

		Tableau t = new Tableau("tab");
		for(int i = 0; i<5; i++)
```

```
            {
                  System.out.println(t.getTab(i));
            }

            ---

            try
            {
                  Tableau t = new Tableau("tab");
            }
            catch(IOException e)
            {
                  System.out.println(e.getMessage());
            }
      }
}
```

ErrTableau.java :

```
public class ErrTableau extends Exception
{
      public ErrTableau(int err)
      {
            super("Taille de tableau invalide : " + err);
            printStackTrace();
      }

      public ErrTableau(String err)
      {
            super(err);
            printStackTrace();
      }
}
```