

**Compte rendu TP 4 :**

**GC, Exceptions et Assertions**

Atelier Java

**Travail réalisé par :**

Mohamed Aziz Bellaaj

**Groupe :** GL 2/2

I)Garbage Collection

a) Introduction

Garbage Collection est le mécanisme dans Java qui permet de libérer les resources mémoires (les objets) qui ne sont plus utilisés, donc ne possédant pas des références sous cette objet. On peut surcharger la méthode finalize() pour qu’un objet puisse exécuter un bout de code avant d’être détruit.

b) Conception

1)Garbage

Ce programme fait appelle au garbage collector à travers l’instruction System.gc() afin de détruire l’objet crée de type Garbage qui ne possède aucune sur lui. Une fois détruit, l’objet exécute sa propre méthode finalize et affiche « Bye ».

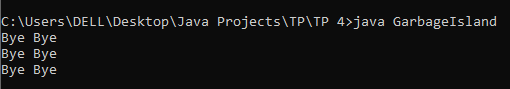
|  |
| --- |
| class Garbage{  public void finalize(){  System.out.println("Bye");  }  public static void main(String[]args){  Garbage i= new Garbage();  i=null;  System.gc();  }  } |



2)GarbageIsland

Un objet qui possède des références à d’autres objets mais qui eux-mêmes ne possèdent pas des références sont susceptibles d’être libérés par le Garbage Collector. Ce bout de code illustre exactement ce concept.

|  |
| --- |
| Public class GarbageIsland{  GarbageIsland n;  public static void main(String [] args){  GarbageIsland i2 = new GarbageIsland();  GarbageIsland i3 = new GarbageIsland();  GarbageIsland i4 = new GarbageIsland();  i2.n = i3;  i3.n = i4;  i2 = null;  i3 = null;  i4 = null;  while(true);  System.gc();  }  public void finalize(){  System.out.println("Bye Bye");  }  } |



II)Les exceptions

a) Introduction

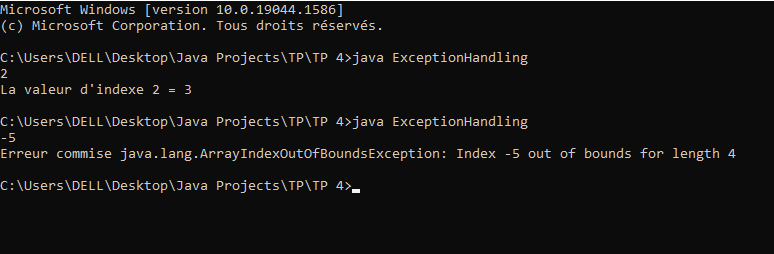
Dans Java, on peut gérer les exceptions en entourant le code susceptible de générer une erreur avec un block try. Si une exception est détectée, on peut exécuter un bout de code dans le block catch.

b) Conception de programme

1)ExceptionHandling

On crée un tableau tab qui contient 4 éléments puis on demande de l’utilisateur lequel il veut afficher en précisant l’indexe de cet élément. Si il choisit un indexe correct (entre 0 et 3), l’affichage déroulera comme prévu. Si non, On affiche un message contenant l’erreur commise.

|  |
| --- |
| import java.util.Scanner;  public class ExceptionHandling{  public static void main(String [] args){  int[] tab ={1,2,3,4};  Scanner scan= new Scanner(System.in);  try{  int x=scan.nextInt();  System.out.println("La valeur d'indexe "+x+" = "+tab[x]);  }  catch(Exception e){  System.out.println("Erreur commise " + e.toString());  }    }  } |



III)Les Assertions

a) Introduction

Les assertions sont utilisées pour vérifier une condition comme vraie. Si la condition est fausse, une erreur sera générée. Les assertions disparaissent quand le programme est déployé. On exécute le programme avec les assertions activés en utilisant la commande java -ea « nom du programme »

b) Conception

Le programme ci-dessous demande de l’utilisateur de saisir son âge. Si son âge est positif, on va l’afficher dans un message. Si non, une erreur sera produite.

|  |
| --- |
| import java.util.Scanner;  public class Assertions{  public static void main(String [] args){  Scanner scan = new Scanner(System.in);  int age = scan.nextInt();  assert(age>0);  System.out.println("Mon age est "+ age);  }  } |

