Compte Rendu – Projet Hanoi
CHIR Corentin – IT 4A SQR

Question 1 :

Les classes d'équivalence pour la méthode « empiler » sont les suivantes :

Paramètre d'entrée	Classe Valide	Classe Invalide
Diamètre de « d » si la tour est vide (d = diamètre du disque)	[1 ; +oo[	]-oo ; 0]
Diamètre de « d » si la tour n'est pas vide (s = disque au sommet)	[1 ; s.d [(Le disque doit etre plus petit que le sommet)	]-oo ; 0] U [s.d ; +oo[

Tableau 1 – Classes d'équivalence

Question 2 :

Les classes d'équivalence par approche aux limite, pour la méthode « empiler » sont les suivantes :

Paramètre d'entrée	Classe Valide	Classe Invalide
Diamètre de « d » si la tour est vide (d = diamètre du disque)	this.d > 0	this.d <= 0
Diamètre de « d » si la tour n'est pas vide (s = disque au sommet)	this.d > 0 and this.d < s.d and taille() < hauteurMax	this.d =< 0 or this.d => s.d or taille() >= hauteurMax

Tableau 2 – Approche aux limites

Question 3 :

Le graphe de flot de contrôle de la fonction « empiler » est le suivant :

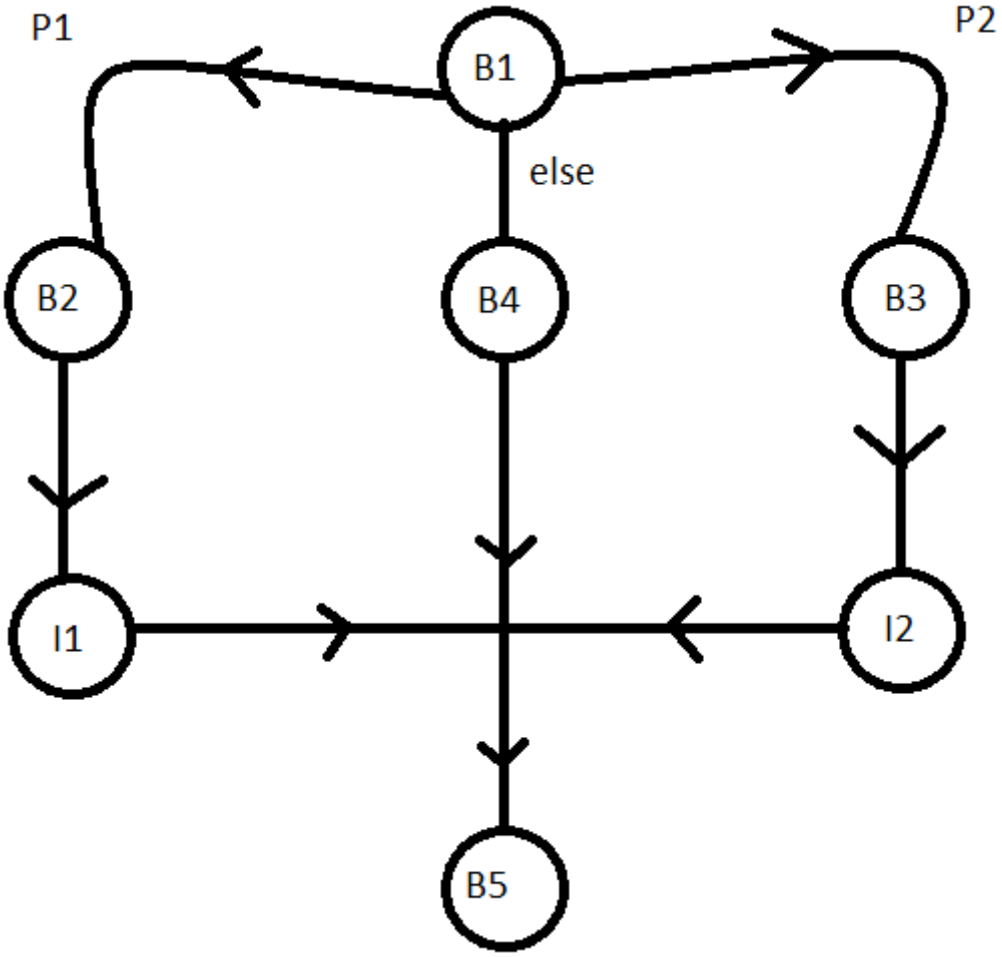

Figure 1 – Diagramme de flot de contrôle de la méthode « empiler »

Ce diagramme peut être transposer dans cette forme alégebrique :
Soit S l'ensemble des chemins de contrôle du graphe « Figure 1 ».

$$S = B1B2I1B5 + B1B4B5 + B1B3I2B5$$

$$S = B1 . (B2I1 + B4 + B3I2) . B5$$

Question 4 :

Pour couvrir l'ensemble des instructions, c'est à dire tester tous les passages du graphe de flot de contrôle, les données de test dont nous avons besoin sont le disque en lui-même, son diamètre et la hauteur de la tour.

Question 5 :

Ces données couvrent bien tous les arcs du graphe, car :

- En prenant en compte le disque, on observe si il existe ou pas. Si il existe, on peut rentrer dans les boucles, et si il n'existe pas, une erreur empechera de faire les opérations.
  On influe donc sur toutes les branches du graphe, c'est donc une donnée clé.
- En prenant en compte le diamètre du disque, on peut également choisir quelle branche du graphe on parcourt.
  Une mauvaise taille nous mène à B4 , et une taille convenable couplée à une taille de tour convenable nous mène à B3.
  Il faut donc y avoir accès pour accéder à tous les chemins, c'est une donnée clé.
- En prenant en compte la taille de la tour, nous avons accès à toutes les branches du graphe.
  Une taille à 0 indique une tour vide, on ira donc dans la branche commencant par B2.
  Une taille supérieure à la hauteur maximale nous mène à la branche de B4, et une taille convenable couplée à un diamètre convenable nous mène à la branche de B3.
  Il s'agit donc également d'une donnée clé.

Question 8 :

Dans un premier temps, nous allons tester le bon fonctionnement de la méthode « compareTo » des disques.
Voici les tests effectués dans mon programme (Figure 2) :

```
//Test des disques
Disc dOr = new Disc(5);
Disc moby = new Disc(20);
Disc oFever = new Disc(50);

dOr.compareTo(moby);
dOr.compareTo(oFever);

moby.compareTo(dOr);
moby.compareTo(oFever);

oFever.compareTo(dOr);
oFever.compareTo(moby);
```

Figure 2 – Test de la fonction « compareTo » de Disque

Et le résultat (Figure 3) :

```
Ce disque peut être déplacé (plus petit que celui visé)
Ce disque peut être déplacé (plus petit que celui visé)
Ce disque ne peut pas être déplacé (plus grand que celui visé)
Ce disque peut être déplacé (plus petit que celui visé)
Ce disque ne peut pas être déplacé (plus grand que celui visé)
Ce disque ne peut pas être déplacé (plus grand que celui visé)
```

Figure 3 – Résultat des tests

On remarque qu'on ne peut pas déplacer des disques plus grand sur un disque plus petit sans avoir un message d'erreur nous indiquant clairement le problème.

Question 7 :

Voici les tests effectués pour le fonctionnement des tours (Figure 4) :

```
//Test des tours
Tower Babel = new Tower();

System.out.println(Babel.taille());
Babel.empiler(dOr);
System.out.println(Babel.taille());
Babel.empiler(oFever);
System.out.println(Babel.taille());

Babel.depiler();
System.out.println(Babel.taille());

System.out.println("");

Babel.empiler(oFever);
Babel.empiler(moby);
Babel.empiler(dOr);
System.out.println(Babel.taille());
System.out.println(Babel.sommet());
System.out.println(Babel.estVide());
System.out.println(Babel.estPleine());
```

Figure 4 – Test des fonctions de Tour

Et leurs résultats (Figure 5) :

```
0
1
Ce disque ne peut pas être déplacé (plus grand que celui visé)
1
0

Ce disque peut être déplacé (plus petit que celui visé)
Ce disque peut être déplacé (plus petit que celui visé)
3
Ce disque a un rayon de 5cm.
La tour n'est pas vide.
false
La tour est pleine.
true
```

Figure 5 – Résultat des test pour les fonctions de Tour

Comme on peut voir, on ne peut toujours pas empiler des disques plus grands sur des plus petits.
De plus, ma technique pour dépiler, qui consiste à remplacer par un disque nul, a été perfectionné et fonctionne correctement comme on peut le voir après avoir enlever le seul disque, puis rajouté 3 disques et voir un résultat de 3 lorsque l'on demande la taille de la tour.
Les fonctions « estPleine » et « estVide » sont également opérationnelles.

Ces différents tests sont accessibles dans le code mis à votre disposition, dans mon répertoire « HanoiTowerOfDoom », dans la classe Main.