```
8       //[...]
9   }
```

Question 1

Nous n'avons pas encore le code !

Il faut déterminer les **classes d'équivalence** à considérer pour la méthode *empiler*, en remplissant le tableau suivant :

Paramètre d'entrée	Classe valide	Classe invalide
Diamètre de d si la tour est vide (d = diamètre du disque)	$d \in]0; MAX\_VALUE]$	$d \leq 0 \cup d > MAX\_VALUE$
Diamètre de d si la tour n'est pas vide (s = disque au sommet)	$d \in]0; s[$	$d \leq 0 \cup d \geq s$

Tableau 1 Classes d'équivalence

Question 2

Déterminer maintenant, par une *approche aux limites*, les données de tests à produire pour la méthode *empiler*, en remplissant le tableau suivant :

Paramètre d'entrée	Classe valide	Classe invalide
Diamètre de d si la tour est vide (d = diamètre du disque)	$d = 1$ $d = MAX\_VALUE$	$d = 0$ $d = MAX\_VALUE+1$
Diamètre de d si la tour n'est pas vide (s = disque au sommet)	$d = 1$ $d = s-1$	$d = 0$ $d = s$

Tableau 2 approche aux limites

« *Rappel : le test des valeurs limites n'est pas vraiment une famille de sélection de test, mais une tactique pour améliorer l'efficacité des données de test (DT) produites par d'autres familles. Les erreurs se nichent généralement dans les cas limites, d'où le fait qu'on teste principalement les valeurs aux limites des domaines ou des classes d'équivalence...* »

Sélection de tests boîte blanche (tests structurels)

Soit le programme en langage Java suivant pour la classe **Tour** des Tours de Hanoï :

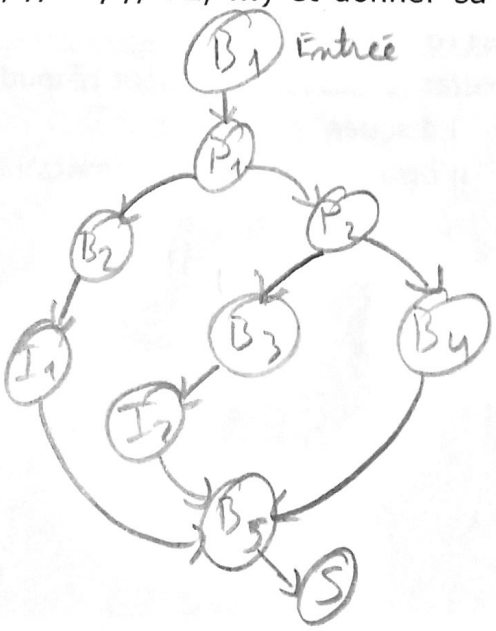
```
1   public class Tour {
2
3       Queue<Disque> disques=new ArrayDeque<Disque>();
4
5       public int diam(){
6          return this.disques.element().d;
7       }
8
9       public int taille() {
```

```
10          return disques.size();
11      }
12
13      boolean empiler(Disque d){
14          boolean res=false; // B1
15          if(this.disques.isEmpty()){ // P1
16              this.disques.offer(d); // B2
17              res=true; // I1
18          }
19          else{
20              if( (diam()>d.d) && (taille()<hauteurMax) ){{ // P2
21                  this.disques.offer(d); // B3
22                  res=true; // I2
23              }
24              else{
25                  res=false; // B4
26              }
27          }
28          return res; // B5
29      }
30  }
```

Question 3

Dessiner le graphe de flot de contrôle correspondant à ce programme (en se servant des annotations indiquées en commentaires : // B1, // P1, // B2, // I1, // P2, ...) et donner sa forme algébrique :

> $\{B_1 ; P_1 ; B_2 ; I_1 ; B_5\}$
>
> $\{B_1 ; P_1 ; P_2 ; B_3 ; I_2 ; B_5\}$
>
> $\{B_1 ; P_1 ; P_2 ; B_4 ; B_5\}$
>
> $B_1 P_1 B_2 I_1 B_5 + B_1 P_1 P_2 B_3 I_2 B_5 + B_1 P_1 P_2 B_4 B_5$
>
> $= B_1 P_1 B_5 (B_2 I_1 + P_2 B_3 I_2 + P_2 B_4)$
>
> $= B_1 P_1 B_5 (B_1 I_2 + P_2 (B_3 I_2 + B_4))$



Question 4

Trouver les données de test minimales pour couvrir toutes les instructions (couverture de tous les nœuds du graphe de flot de contrôle)

Question 5

Ces données de test assurent-elles la couverture de tous les arcs du graphe ? Sinon ajouter de nouvelles données de test pour couvrir tous les arcs.

Question 6

La ligne ci-dessous représente une condition composée (deux expressions booléennes reliées par un ET logique) :

```
1   (diam()>d.d) && (taille()<hauteurMax)
```