TP_B

SORTIE DU DU PROGRAMME:

[05h 14mn 34,184s] Début du programme.
Simplet veut la ressource.
Grincheux veut la ressource.
Joyeux veut la ressource.
Atchoum veut la ressource.
Dormeur veut la ressource.
Timide veut la ressource.
Prof veut la ressource.
Simplet accède à la ressource.
[05h 14mn 34,191s]Simplet a un accès (exclusif) à Blanche-Neige.
[05h 14mn 36,193s] Simplet s'apprête à quitter Blanche-Neige.
Simplet relâche la ressource.
Grincheux accède à la ressource.
[05h 14mn 36,193s]Simplet a terminé!
[05h 14mn 36,193s]Grincheux a un accès (exclusif) à Blanche-Neige.
Simplet veut la ressource.
[05h 14mn 38,195s] Grincheux s'apprête à quitter Blanche-Neige.
Grincheux relâche la ressource.
Joyeux accède à la ressource.
[05h 14mn 38,195s]Grincheux a terminé!
Grincheux veut la ressource.
[05h 14mn 38,195s]Joyeux a un accès (exclusif) à Blanche-Neige.
[05h 14mn 39,192s] Interruption 7 nains.
[05h 14mn 39,192s]Timide a terminé!
[05h 14mn 39,192s]Grincheux a terminé!
[05h 14mn 39,192s]Simplet a terminé!
[05h 14mn 39,192s]Atchoum a terminé!
[05h 14mn 39,192s]Prof a terminé!
[05h 14mn 39,192s] Joyeux s'apprête à quitter Blanche-Neige.
Joyeux relâche la ressource.
[05h 14mn 39,192s]Dormeur a terminé!
[05h 14mn 39,193s]Joyeux a terminé!
[05h 14mn 39,192s] Joyeux s'apprête à quitter Blanche-Neige.
Joyeux relâche la ressource.
[05h 14mn 39,192s]Dormeur a terminé!
[05h 14mn 39,193s]Joyeux a terminé!
[05h 14mn 39,193s] Fin des nains.

```java
// -*- coding: utf-8 -*-

import java.util.ArrayList;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.concurrent.*;

public class SeptNains {
    static private SimpleDateFormat sdf =
        new SimpleDateFormat("hh'h 'mm'mn 'ss','SSS's'");

    public static void main(String[] args) throws InterruptedException {
        Date début = new Date(System.currentTimeMillis());
        System.out.println("[" + sdf.format(début) + "] Début du programme.");

        final BlancheNeige bn = new BlancheNeige();
        final int nbNains = 7;
        final String noms [] = {"Simplet", "Dormeur",  "Atchoum", "Joyeux",
                                "Grincheux",
                                "Prof", "Timide"};
        final Nain nain [] = new Nain [nbNains];
        for(int i = 0; i < nbNains; i++) nain[i] = new Nain(noms[i],bn);
        for(int i = 0; i < nbNains; i++) nain[i].start();

        /*Question 1
            java.lang.InterruptedException: sleep interrupted
            at java.base/java.lang.Thread.sleep(Native Method)
            at Nain.run(SeptNains.java:71)
        */
        try {
            Thread.sleep(5000);
        } catch(InterruptedException exception) {
            exception.printStackTrace();
        } finally {
            System.out.println("[" + sdf.format(new Date(System.currentTimeMillis()))
                 + "] Interruption 7 nains.");
            for(int n = 0; n < nbNains; n++){
                nain[n].interrupt();
            }
        }

        /*Question 3*/
        for(int n = 0; n < nbNains; n++){
            nain[n].join();
        }

        System.out.println("[" + sdf.format(new Date(System.currentTimeMillis()))
            + "] Fin des nains.");
    }
}

class BlancheNeige {

    /* file d'attente pour un partage équilibré pour l'accès à la ressource
    * http://blog.paumard.org/cours/java-api/chap05-concurrent-queues.html
    */
    private BlockingQueue<Nain> file_acces = new ArrayBlockingQueue<Nain>(7);


    private volatile boolean libre = true; // Initialement, Blanche-Neige est libre.
```

```
61      public synchronized void requérir () {
62
63          //le nain courant est ajouté à la file car il souhaite un accès à la
    ressource
64          Nain threadCourant = (Nain) Thread.currentThread();
65          file_acces.add(threadCourant);
66
67          System.out.println("\t" + threadCourant.getName()
68                             + " veut la ressource.");
69      }
70
71      public synchronized void accéder () throws InterruptedException {
72
73          Nain threadCourant = (Nain) Thread.currentThread();
74
75          /* il faut maintenant vérifier que Blanche-Neige est libre
76          * pour donner accès au nain s'il est premier dans la file de priorité
77           https://www.geeksforgeeks.org/arrayblockingqueue-peek-method-in-java/
78            (ref. peek())
79          */
80
81          while ( ! libre || !threadCourant.equals(file_acces.peek())) {
82              wait(); // Le nain s'endort sur l'objet bn
83          }
84          libre = false;
85          System.out.println("\t" + Thread.currentThread().getName()
86                             + " accède à la ressource.");
87      }
88
89      public synchronized void relâcher () {
90          System.out.println("\t" + Thread.currentThread().getName()
91                             + " relâche la ressource.");
92          libre = true;
93
94          //le nain relâche la ressource, il faut donc le retirer de la file d'attente
95          file_acces.poll();
96          notifyAll();
97      }
98  }
99
100 class Nain extends Thread {
101     private BlancheNeige bn;
102
103     static private SimpleDateFormat time =
104         new SimpleDateFormat("hh'h 'mm'mn 'ss','SSS's'");
105
106     public Nain(String nom, BlancheNeige bn) {
107         this.setName(nom);
108         this.bn = bn;
109     }
110
111     public void run() {
112         /*Si le nain non interrompu, il peut avoir accès à la ressource*/
113         while(!isInterrupted()){
114             try {
115                 bn.requérir();
116                 bn.accéder();
117                 System.out.println("[" + time.format(
118                         new Date(System.currentTimeMillis())) + "]"
119                         + getName()
```

```
                        + " a un accès (exclusif) à Blanche-Neige.");

                try { sleep(2000);
                } catch (InterruptedException exception1) {
                    long temps_interruption = System.currentTimeMillis();
                    long temps = System.currentTimeMillis() - temps_interruption;

                        try { sleep(temps);
                        } catch (InterruptedException exception2) {
                            this.interrupt();
                        } finally { this.interrupt(); }

                    } finally {
                    System.out.println("[" + time.format(
                        new Date(System.currentTimeMillis())) + "] "
                        + getName()
                        + " s'apprête à quitter Blanche-Neige.");
                    bn.relâcher(); }
            } catch(InterruptedException exception3) {
                this.interrupt();
            }

            //interruption nains, message de fin
            System.out.println("[" + time.format(
                new Date(System.currentTimeMillis()))
                + "]" + getName()
                + " a terminé!");
        }
    }
}

```