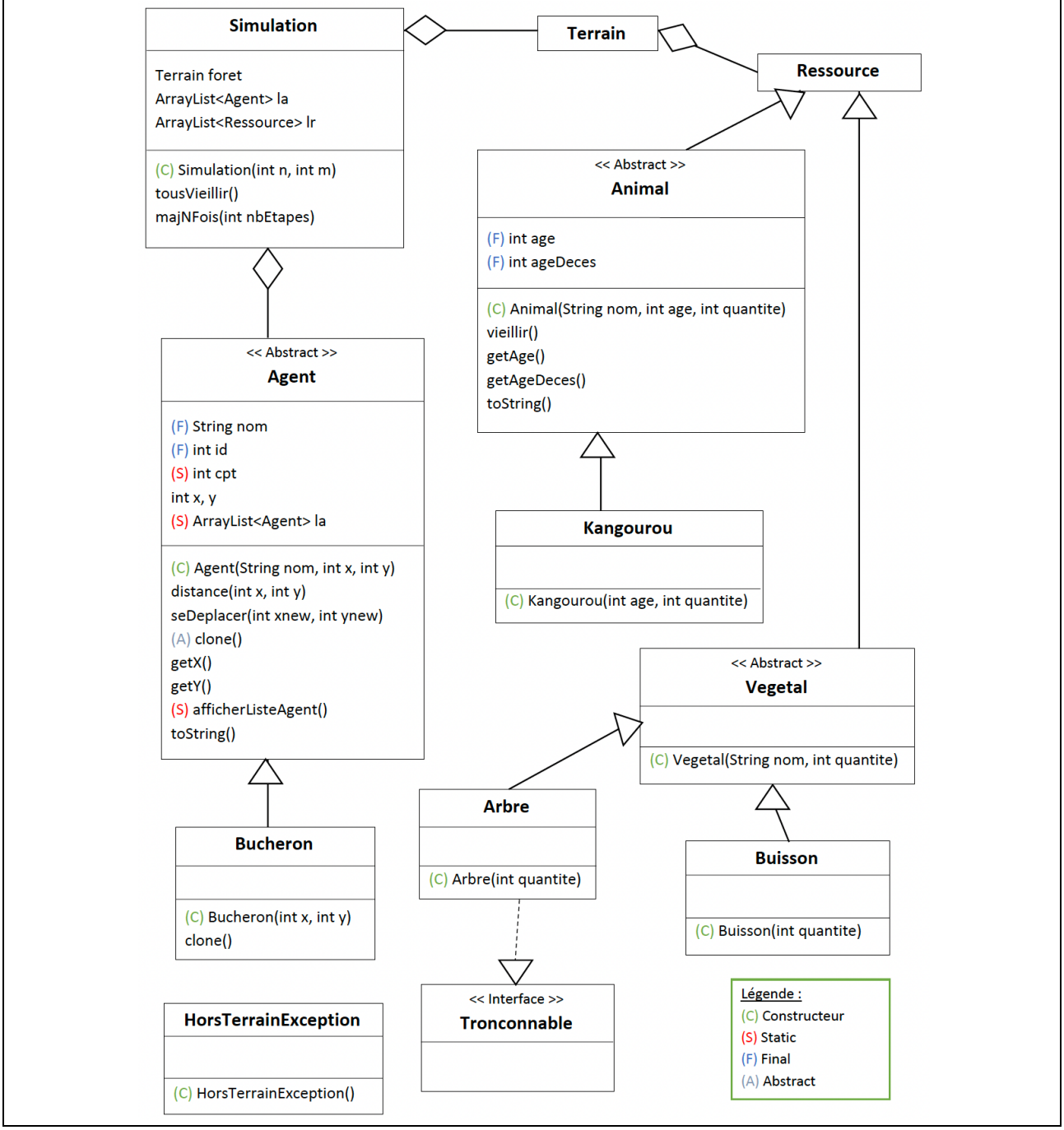
Projet LU2IN002 - 2022-2023

Numéro du groupe de TD/TME : 8

Nom : SONG	*Nom :* BENYAHIA
Prénom : Michelle	*Prénom :* Ilyas
N° étudiant : 21106878	*N° étudiant :* 21118889

Thème choisi (en 2 lignes max.)
Simulation d'une forêt (terrain) contenant des animaux (ressource non modifiable par les agents) et des végétaux, dont les arbres sont entretenus par des bucherons (agents).

Schéma UML des classes vision fournisseur (dessin "à la main" scanné ou photo acceptés)



Checklist des contraintes prises en compte:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une ArrayList	Agent, Simulation
Classe avec membres et méthodes statiques	Agent
Classe abstraite et méthode abstraite	Animal, Vegetal, Agent
Interface	Tronconnable
Classe avec un constructeur par copie ou clone()	Agent, Bucheron
Définition de classe étendant Exception	HorsTerrainException
Gestion des exceptions	Agent, Simualtion

Présentation brève de votre projet (max. 10 lignes) : texte libre expliquant en quoi consiste votre projet.
Notre projet simule une forêt à la taille maximale (indiqué dans Terrain.class : 20x20) dans laquelle on place des ressources aléatoirement, dont le type est animal ou végétal. Ce terrain forestier évolue à l'appel de la méthode de mise à jour du terrain en un nombre fini de fois. En effet, durant la mise à jour, les bucherons (non visibles et non mémorisés sur le terrain) se déplacent, puis en cas de rencontre avec des végétaux « tronçonnable », les coupent puis vident la case. Les ressources animales eux, évoluent sans l'intervention des agents, par vieillissement naturel, en leur attribuant un âge de décès aléatoire. L'affichage dans le terminal montre l'évolution de la forêt au bout de n mises à jour, c'est-à-dire la répartition des ressources.

Classe Agent :

```
import java.util.ArrayList;

public abstract class Agent {
	public final String nom;
	public final int id;
	private static int cpt = 0;
	protected int x, y;
	protected static ArrayList<Agent> la = new ArrayList<Agent>();

	public Agent(String nom, int x, int y) {
	this.nom = nom;
	this.x = x;
	this.y = y;
```

```
        id = cpt;
        cpt++;
        la.add(this);
        }

        public double distance(int x, int y) {
                return Math.sqrt((this.x - x) * (this.x - x) + (this.y - y) * (this.y - y));
        }

        public boolean seDeplacer(int xnew, int ynew) throws HorsTerrainException {
                try {
                        if (((xnew >= 0) && (xnew < Terrain.NBLIGNESMAX)) && ((ynew >= 0)
&& (ynew < Terrain.NBCOLONNESMAX))) {
                                for (Agent a : la) {
                                        if ((a.getX() == xnew) && (a.getY() == ynew)) {
                                        return false;
                                        }
                                }

                                x = xnew;
                                y = ynew;
                        } else {
                                throw new HorsTerrainException();
                        }

                } catch (HorsTerrainException e) {
                        int x = (int)(Math.random() * (Terrain.NBLIGNESMAX));
                        int y = (int)(Math.random() * (Terrain.NBCOLONNESMAX));
                        for (Agent ag : la) {
                                while ((ag.getX() == x) && (ag.getY() == y)) { // jusqu'à qu'on
trouve des coordonnées valides (dans le terrain)
                                        x = (int)(Math.random() * (Terrain.NBLIGNESMAX));
                                        y = (int)(Math.random() * (Terrain.NBCOLONNESMAX));
                                }
                        }

                        seDeplacer(x, y);
                }

                return true;
        }

        public abstract Agent clone();

        public int getX() {
                return x;
        }

        public int getY() {
                return y;
        }
```

```
	// test d'affichage

	public static void afficherListeAgent() {
		for (Agent a : la) {
			System.out.println(a.toString());
		}
	}

	public String toString() {
		return String.format("%s [id : %d] en position (%d, %d)", nom, id, x, y);
	}

}
```

Classe Bucheron :

```
public class Bucheron extends Agent {

	public Bucheron(int x, int y) {
		super("Bucheron", x, y);
	}

	public Agent clone() {
		return new Bucheron(getX(), getY());
	}

}
```

Classe Animal :

```
public abstract class Animal extends Ressource {
	private int age;
	private final int ageDeces;

	public Animal(String nom, int age, int quantite) {
		super(nom, quantite);
		this.age = age;
		ageDeces = (int)(Math.random()*50);
	}

	public void vieillir() {
		if ((getX() != -1) && (getY() != -1)) {
			age++;
		}
	}

	public int getAge() {
		return age;
```

```
	}

	public int getAgeDeces() {
		return ageDeces;
	}

	public String toString() {
		return String.format("%s [id : %d quantite : %d age : %d] en position (%d, %d)",
super.type, super.ident, getQuantite(), age, getX(), getY());
	}

}
```

Classe Kangourou :

```
public class Kangourou extends Animal {

	public Kangourou(int age, int quantite) {
		super("Kangourou", age, quantite);
	}

}
```

Classe Vegetal :

```
public abstract class Vegetal extends Ressource {
	public Vegetal(String nom, int quantite) {
		super(nom, quantite);
	}

}
```

Interface Tronconnable :

```
public interface Tronconnable {

}
```

Classe Arbre :

```
public class Arbre extends Vegetal implements Tronconnable {

	public Arbre(int quantite) {
		super("Arbre", quantite);
	}

}
```

Classe Buisson :

```
public class Buisson extends Vegetal {

	public Buisson(int quantite) {
		super("Buisson", quantite);
	}

}
```

Classe HorsTerrainException :

```
public class HorsTerrainException extends Exception {

	public HorsTerrainException() {
		super("Coordonnees en dehors du terrain");
	}

}
```

Classe Simulation :

```
import java.util.*;

public class Simulation {
	private Terrain foret;
	private ArrayList<Agent> la;
	private ArrayList<Ressource> lr;

	public Simulation(int n, int m) throws HorsTerrainException {
		foret = new Terrain(); // creation d'une foret
		lr = new ArrayList<Ressource>();
		la = new ArrayList<Agent>();

		int x = (int)(Math.random() * (Terrain.NBLIGNESMAX));;
		int y = (int)(Math.random() * (Terrain.NBCOLONNESMAX));;

		for (int i = 0; i < m; i++) { // place m ressources aléatoirement
			while (foret.caseEstVide(x, y) == false) {
				x = (int)(Math.random() * (Terrain.NBLIGNESMAX));
				y = (int)(Math.random() * (Terrain.NBCOLONNESMAX));
			}

			int quantite = (int)(Math.random() * 101);

			if (Math.random() < 0.6) {
				Kangourou k = new Kangourou(0, quantite);
				foret.setCase(x, y, k);
				lr.add(k);
			} else {
				if (Math.random() < 0.3) {
```

```
                                        Buisson b = new Buisson(quantite);
                                        foret.setCase(x, y, b);
                                        lr.add(b);
                                } else {
                                        Arbre a = new Arbre(quantite);
                                        foret.setCase(x, y, a);
                                        lr.add(a);
                                }
                        }

                }


                for (int j = 0; j < n; j++) {
                        for (Agent ag : la) {
                                while ((ag.getX() == x) && (ag.getY() == y)) { // genere n agents alé
atoirement
                                        x = (int)(Math.random() * (Terrain.NBLIGNESMAX));
                                        y = (int)(Math.random() * (Terrain.NBCOLONNESMAX));
                                }
                        }

                        Bucheron bu = new Bucheron(x, y);
                        la.add(bu);

                }


                // test de la methode clone

                // System.out.println("Clonage de " + (la.get(0)).nom + " en cours ");
                // System.out.println(la.get(0).toString());
                la.remove(n - 1);
                Bucheron bubu = (Bucheron)((Bucheron)la.get(0)).clone();
                // System.out.println(bubu.toString());
                x = (int)(Math.random() * (Terrain.NBLIGNESMAX + 40));
                y = (int)(Math.random() * (Terrain.NBCOLONNESMAX + 40));

                bubu.seDeplacer(x, y);
                // System.out.println(bubu.toString());
                la.add(bubu);

        }


        public void tousVieillir() {
                for (int i = 0; i < lr.size(); i++) {
                        if (lr.get(i) instanceof Animal) {
                                ((Animal)(lr.get(i))).vieillir();

                        }
                }
```

```
	}

	public void majNFois(int nbEtapes) throws HorsTerrainException {
		System.out.println("Informations sur le terrain:\n" + foret) ;
		foret.affiche(4) ;

		int i = 0;
		while (i < nbEtapes) {
			System.out.println(">>>>>>>> NOUVELLE ETAPE " + i);
			System.out.println("Informations sur le terrain :\n" + foret);

			tousVieillir();

			for (int k = 0; k < lr.size(); k++) {
				if (lr.get(k) instanceof Animal) {
					if (((Animal)(lr.get(k))).getAge() >
(((Animal)(lr.get(k))).getAgeDeces())) { // si l'animal meurt
						if ((((Animal)(lr.get(k))).getX() != -1) &&
((Animal)(lr.get(k))).getY() != -1) { // pour afficher que les animaux morts maintenant
							System.out.println(lr.get(k).toString() + " a tré
passé");
							foret.videCase((lr.get(k)).getX(),
(lr.get(k)).getY());
							(lr.get(k)).initialisePosition(); // met les
positions à (-1, -1)
						}
					}
				}
			}

			for (Agent a : la) { // parcourt la liste d'agents
				int x = (int)(Math.random() * (Terrain.NBLIGNESMAX + 1));
				int y = (int)(Math.random() * (Terrain.NBCOLONNESMAX + 1));

				a.seDeplacer(x, y);

				for (int j = 0; j < lr.size(); j++) { // parcourt la liste de ressources
					if ((a.getX() == (lr.get(j)).getX()) && (a.getY() ==
(lr.get(j)).getY()) && (lr.get(j) instanceof Tronconnable)) {
						if (Math.random() < 0.6) { // les bucherons coupent les
arbres présents sur leur position à une probabilité de 0.6
							foret.videCase((lr.get(j)).getX(),
(lr.get(j)).getY());
							(lr.get(j)).initialisePosition();
							(lr.get(j)).setQuantite(0);
							System.out.println(a.toString() + " a coupé " +
(lr.get(j)).type);
							System.out.println((lr.get(j)).toString());
						}
					}
				}
```

```
                        }

                        i++;
                        System.out.println("Informations sur le terrain:\n" + foret) ;
                        foret.affiche(4);

                        try {
                                Thread.sleep(500);
                        } catch (InterruptedException e) {

                        }

                }
        }

}
```

Classe TestSimulation :

```
public class TestSimulation {
        public static void main(String[] args) throws HorsTerrainException {
                Simulation s = new Simulation(50, 200); // n agents, m ressources
                s.majNFois(70);

        }
}
```