
Forma
JAVA EDITION
Stay strong!
Alexis Ro
Elise
Kerrian
Valider
Passif


LES BASES

Inventory

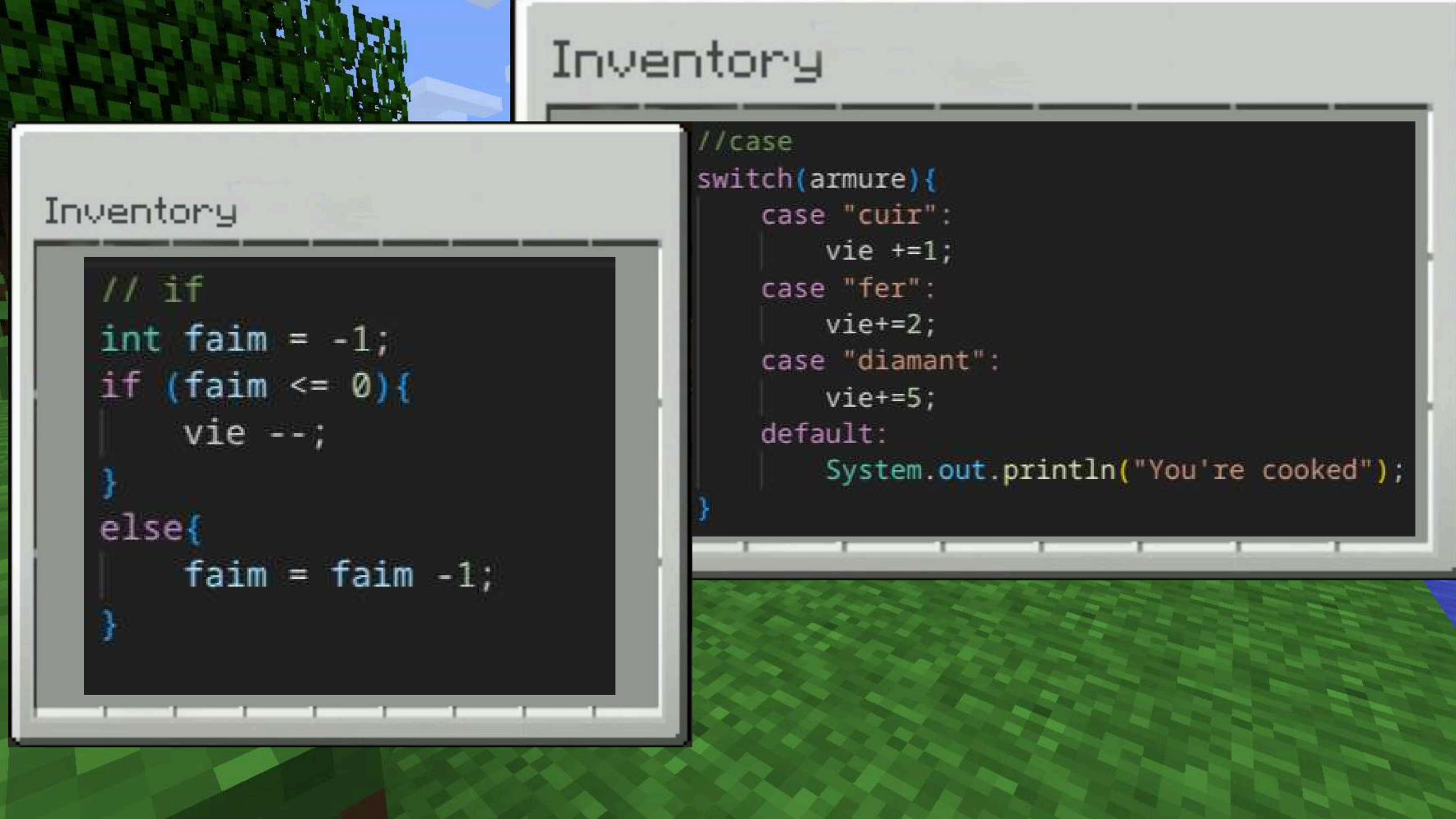
```java
// Les types primitifs
int HeuresDeRevisions = 3;
double zombiesMorts = 1.9e+57d;
float pi = 3.14159265359f;
boolean leJavaVaGagner = false;
char maLettrePreferée = 'A';
long hoursPlayed = 876545678876547L;
short nbDeMorts = 20;
byte  emeraudes = 62;
```

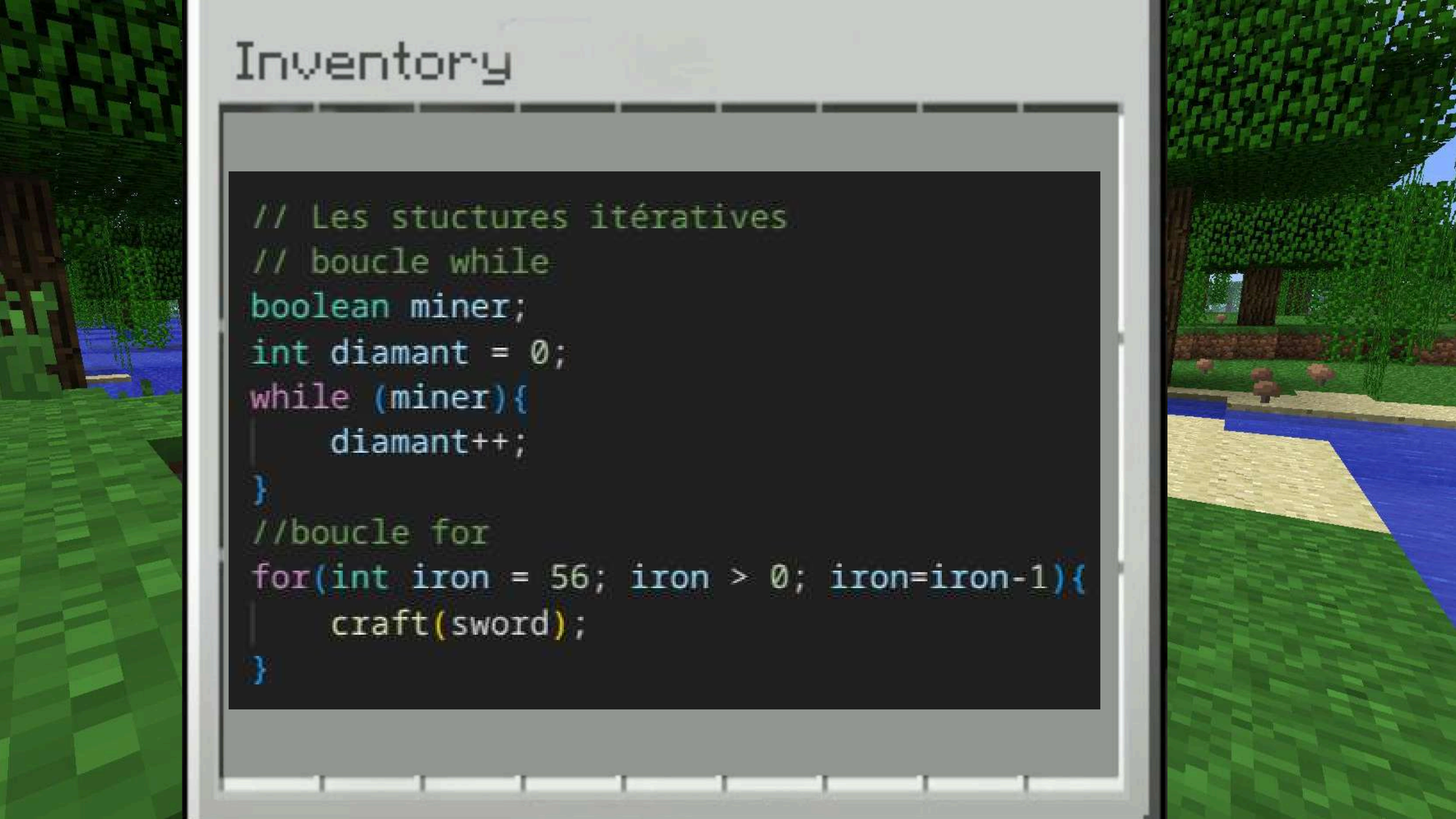
Attention !

String n'est pas un type primitif

Inventory

```java
// Les conversion
Double revisionsToDouble  = (double) 3;
Integer zombiesMortsEntiers  = (int) (double) 1.9e+57d;
Integer piToInteger= (int) 3.14159265359f;
String  leJavaVaGagnerEnString= Boolean.toString(false);
Integer monASCIIprefere = (int) 'A';
Float diamondsToFloat= (float) (long) 34567;
Double nbDeMortsToDouble = (double) (short) 20;
Short emeraudesToShort= (short) 62;
```


Inventory
Inventory
// if
int faim = -1;
if (faim <= 0){
vie --;
}
else{
faim = faim -1;
}
//case
switch(armure){
case "cuir":
vie +=1;
case "fer":
vie+=2;
case "diamant":
vie+=5;
default:
System.out.println("You're cooked");
}


Inventory
// Les stuctures itératives
// boucle while
boolean miner;
int diamant = 0;
while (miner){
diamant++;
}
//boucle for
for(int iron = 56; iron > 0; iron=iron-1){
craft(sword);
}

Inventory

```
//Tableaux
int[] myFavoriteNumbers = new int[300];
String[] myEnemies = new String[500];
```


LES CLASSES

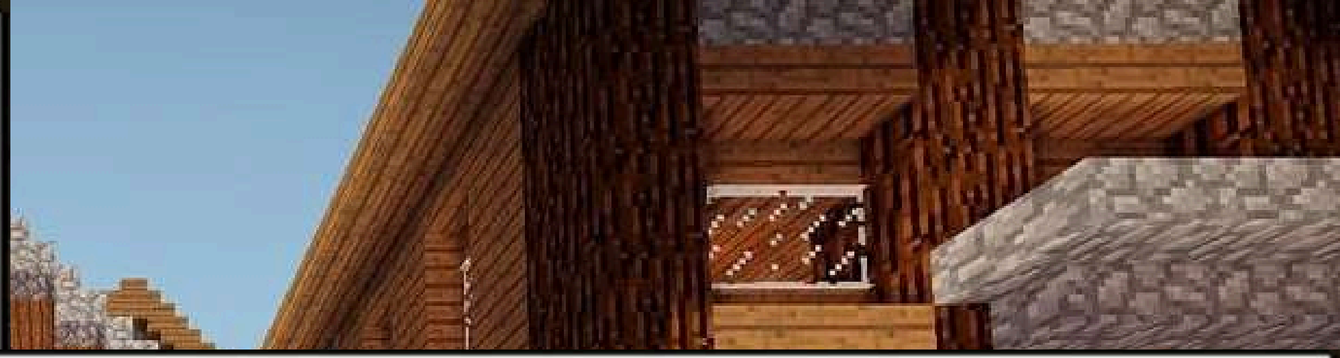

Inventory
Outils.java
src > Outils.java > ...
1 public class Outils {
2 int durability;
3 int damage;
4 String name;}
5
Attributs de la classe

Inventory

```
//Constructeur
Outils(String name, int durability, int damage){
    this.name =name;
    this.durability = durability;
    this.name=name;
}
```

Inventory

```java
// Méthode d'instance d'Outils
public void hit(Enemy enemy){
    enemy.life = enemy.life - damage;
}
```



Inventory

```java
public class Enemy {
    int life;
    String name;
    Enemy(String name){
        this.name = name;
        life = 15;
    }
    //Méthode de classe
    public static boolean isAlive(Enemy enemy){
        return enemy.life >  0;
    }
}
```

Inventory

```java
public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Enemy creeper = new Enemy(name: "Creeper");
        Outils sword = new Outils(name: "sword", durability: 20, damage: 5);
        sword.hit(creeper); //méthode d'instance
        Enemy.isAlive(creeper);//méthode de classe
    }
}
```


Inventory
steve > inventory > J Outils.java > Outils
package steve.inventory;
import enemy.Enemy;
public class Outils {
int durability;
int damage;
String name;

Inventory

```
int durability;
private int damage;
public String name;
```

nventory

Projet

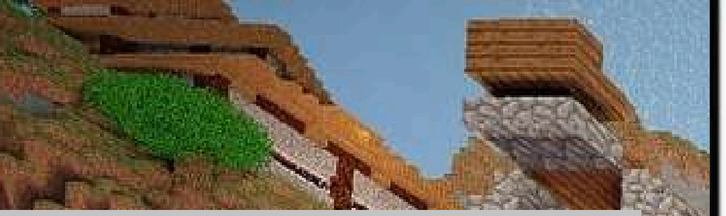
package steve

package enemy

Inventory

```java
public class Sword  extends Outils implements Weapon{
    Sword(){
        super(name: "Sword",durability: 20,damage: 15);
    }
    @Override
    public void hit(Enemy enemy) {
        // TODO Auto-generated method stub
        super.hit(enemy);
    }
}
```

```java
interface Weapon {
    void hit(Enemy enemy);
}
```



Inventory

```java
public class Inventoryofswords {
    int number;
    int damage;
    Sword toUse;
}
```

```java
public class Inventoryofpickaxes{
    int number;
    int damage;
    Outils toUse;
}
```

Inventory

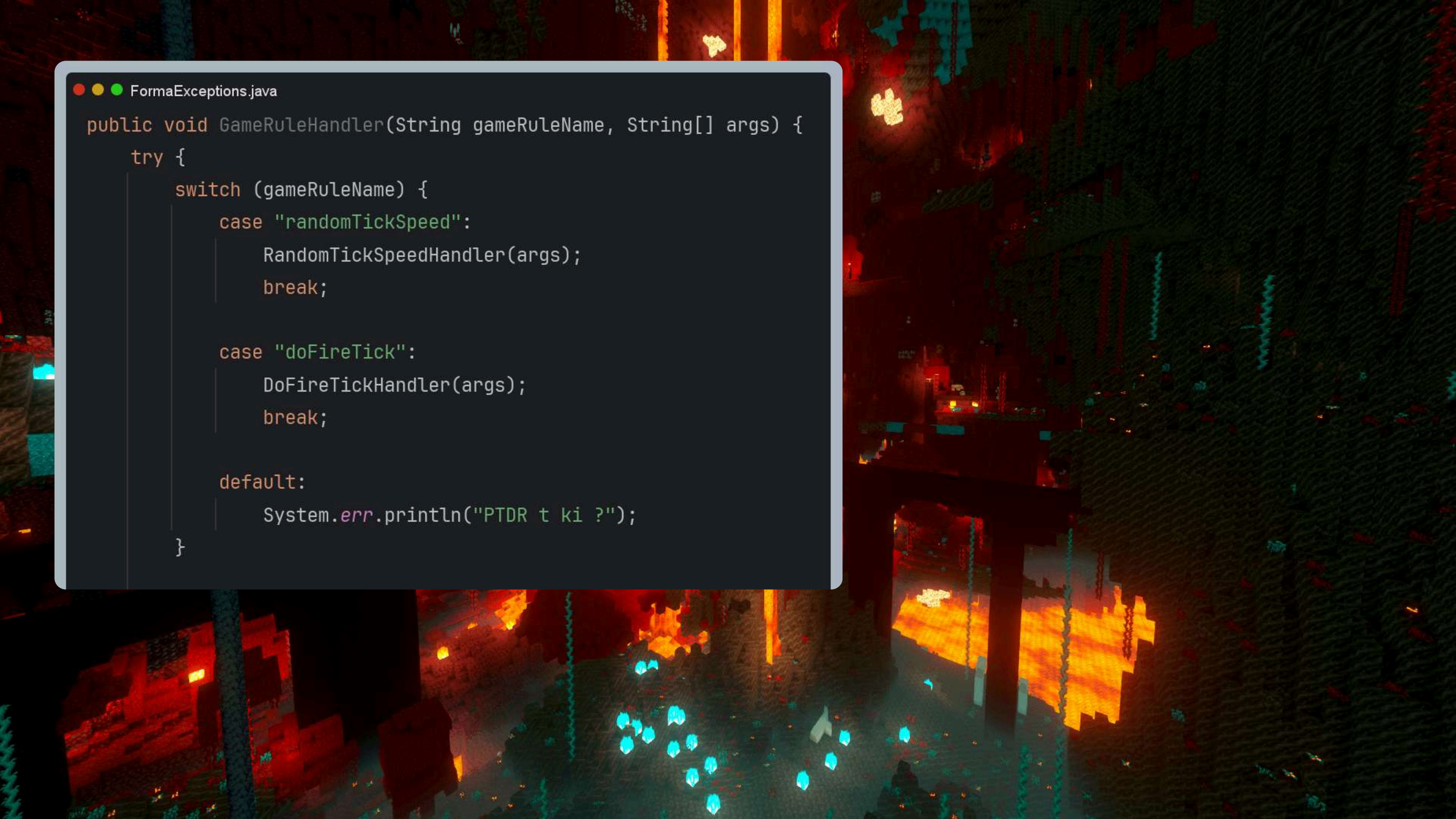
```java
public class Inventoryofthings<T> {
    int number;
    int damage;
    T toUse;
    Inventoryofthings(){
        number=0;
        damage=0;
        toUse = null;
    }
    public void add(T thing){
        number ++;
        toUse = thing;
    }
}
```

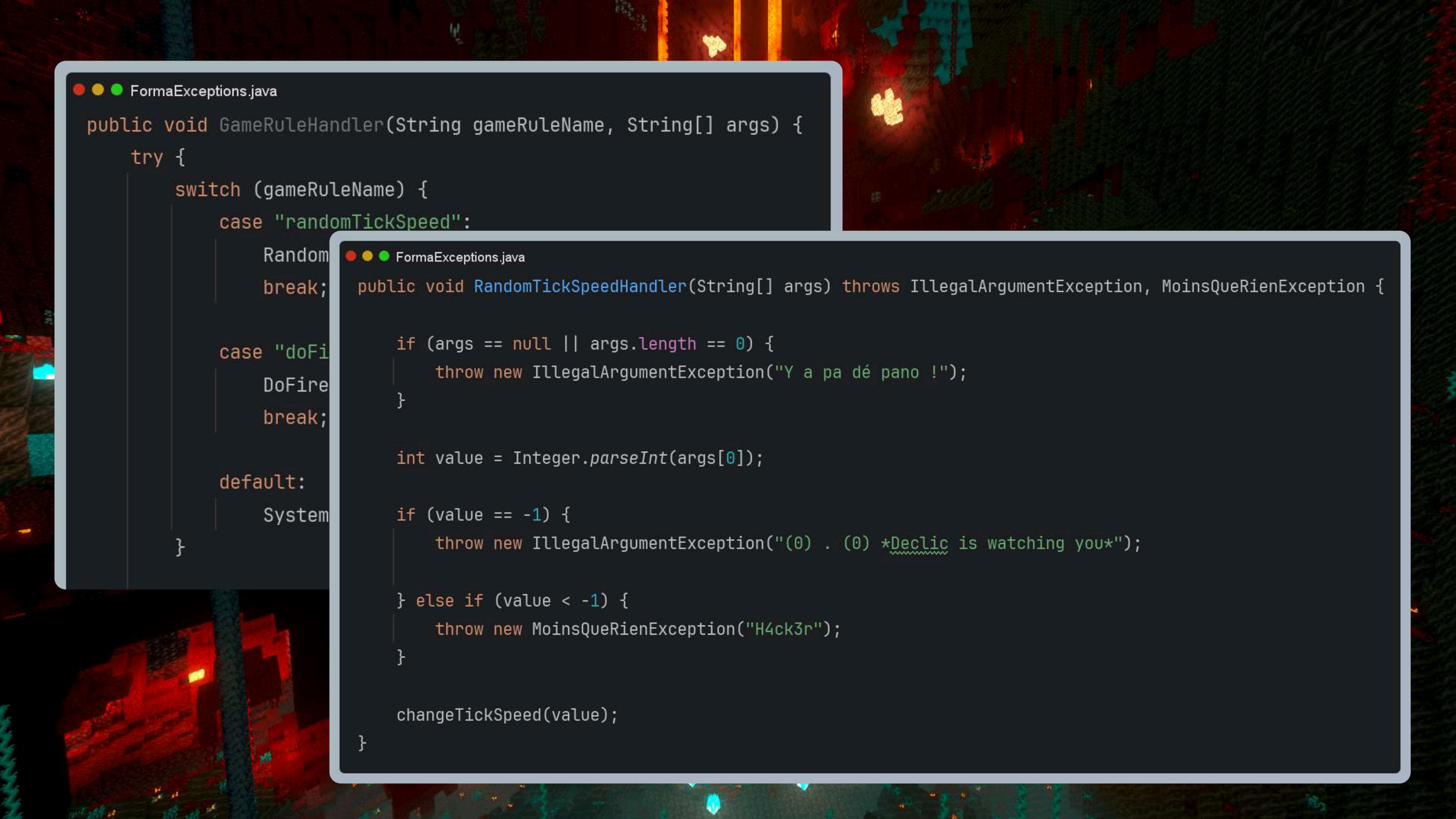
```java
Sword swordWood = new Sword();
Inventoryofthings<Sword> swords = new Inventoryofthings<Sword>();
Inventoryofthings<Enemy> creepers = new Inventoryofthings<>();
swords.add(swordWood);
```

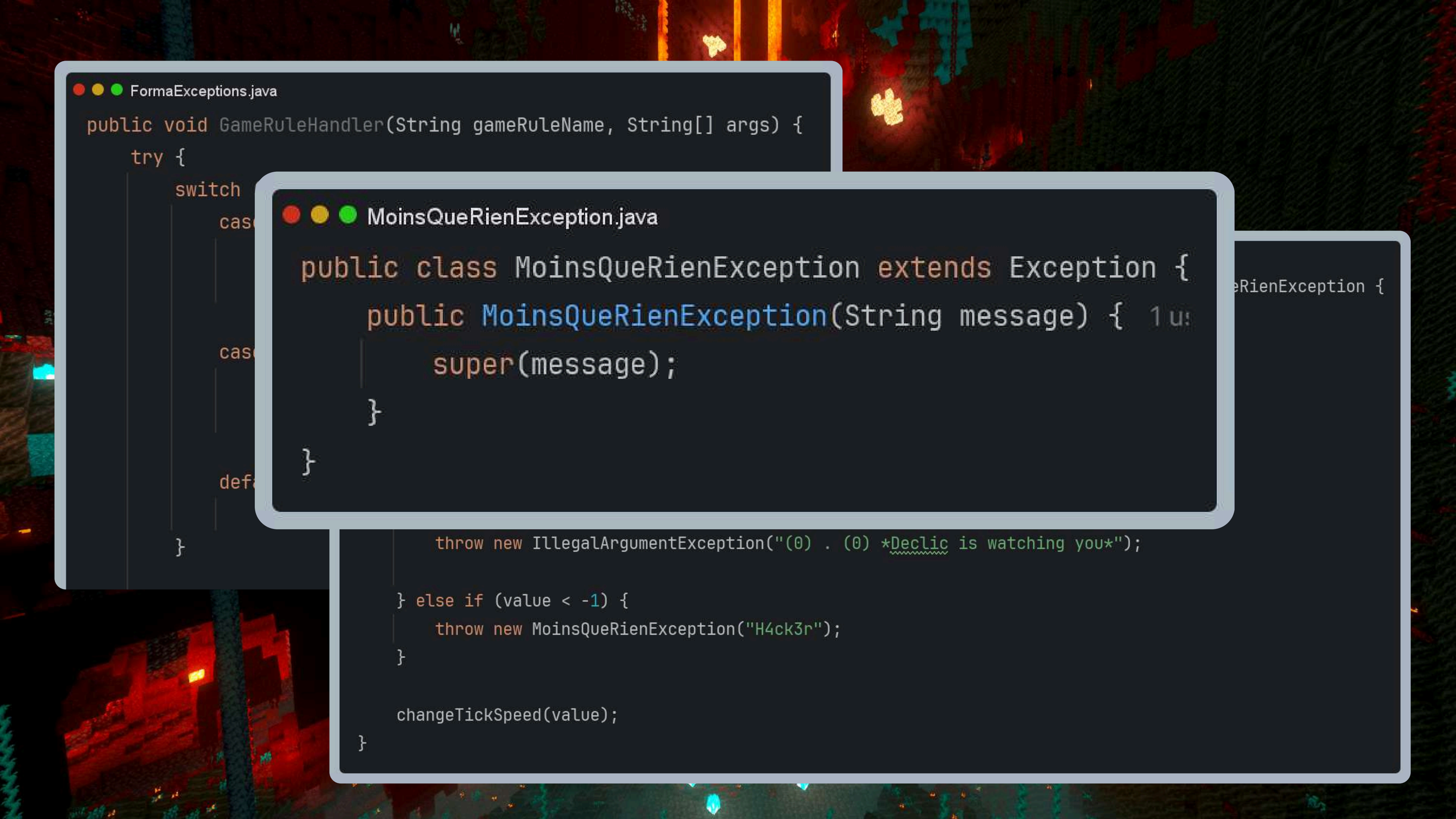

LES EXCEPTIONS

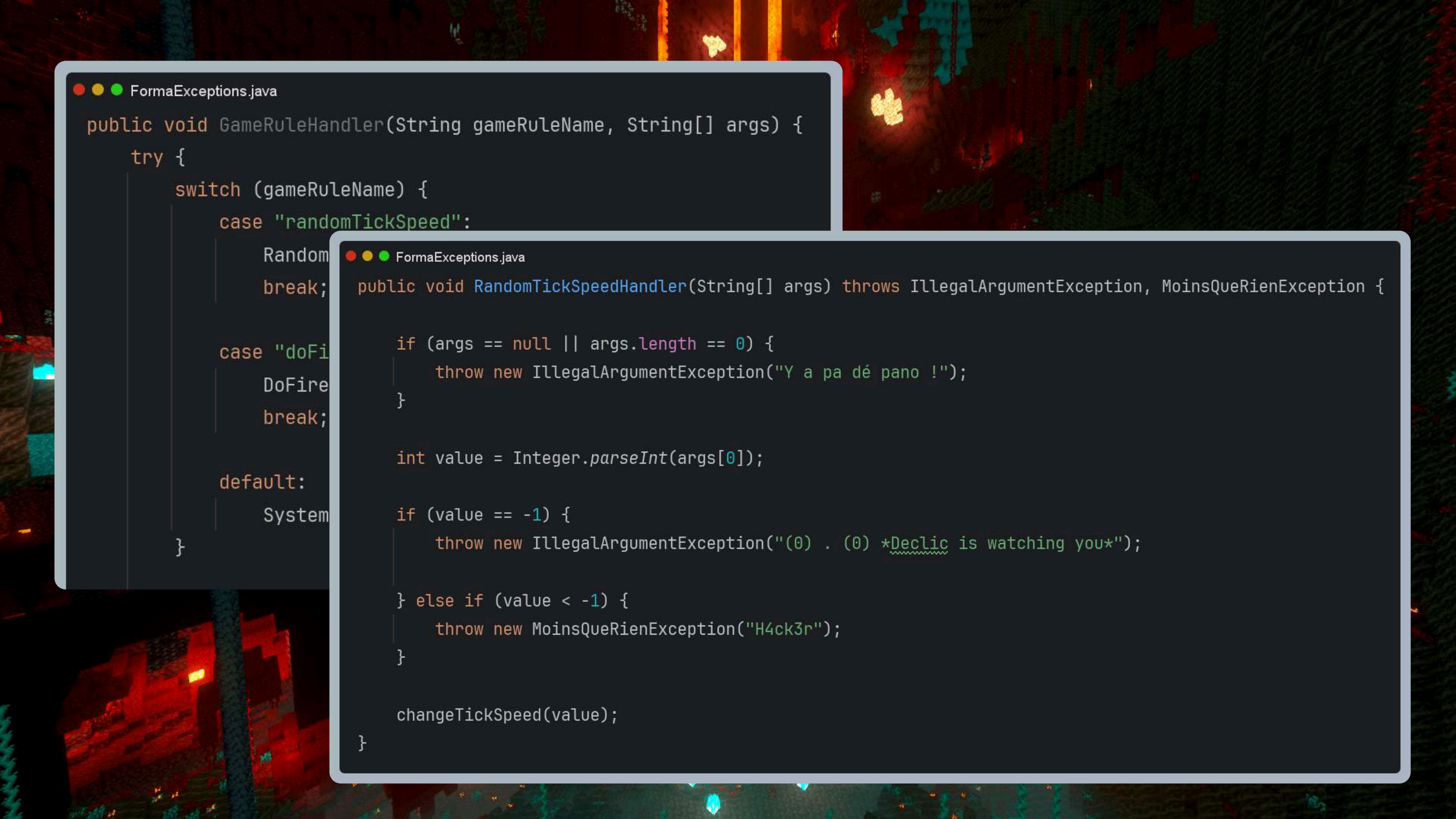

/gamerule

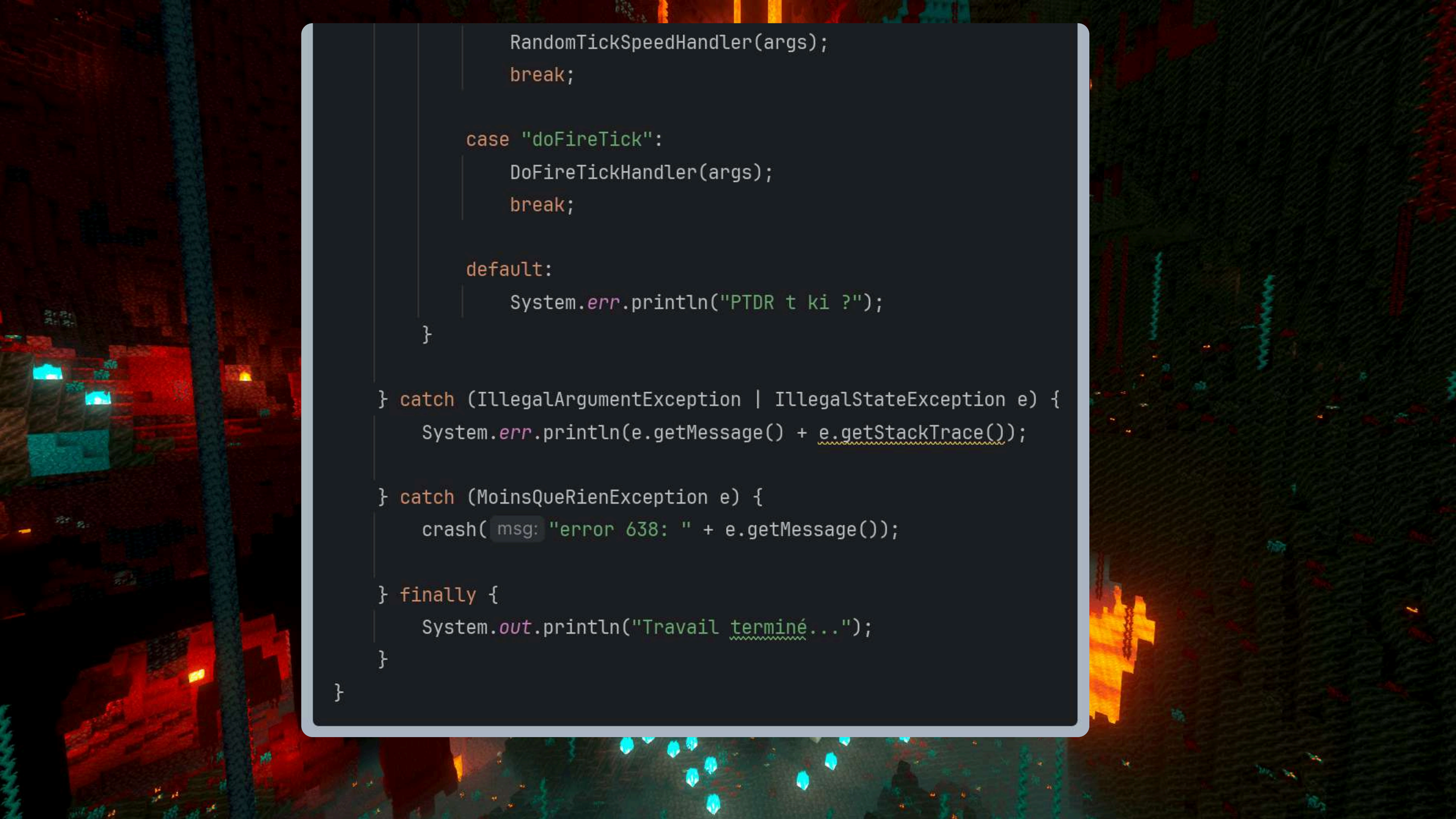

Unknown or incomplete command, see below for error
gamerule<--[HERE]


FormaExceptions.java
public void GameRuleHandler(String gameRuleName, String[] args) {
try {
switch (gameRuleName) {
case "randomTickSpeed":
RandomTickSpeedHandler(args);
break;
case "doFireTick":
DoFireTickHandler(args);
break;
default:
System.err.println("PTDR t ki ?");
}


FormaExceptions.java
public void GameRuleHandler(String gameRuleName, String[] args) {
try {
switch (gameRuleName) {
case "randomTickSpeed":
Random
break;
case "doFi
DoFire
break;
default:
System
}
FormaExceptions.java
public void RandomTickSpeedHandler(String[] args) throws IllegalArgumentException, MoinsQueRienException {
if (args == null || args.length == 0) {
throw new IllegalArgumentException("Y a pa dé pano !");
}
int value = Integer.parseInt(args[0]);
if (value == -1) {
throw new IllegalArgumentException("(0) . (0) *Declic is watching you*");
} else if (value < -1) {
throw new MoinsQueRienException("H4ck3r");
}
changeTickSpeed(value);
}


FormaExceptions.java
public void GameRuleHandler(String gameRuleName, String[] args) {
try {
switch
MoinsQueRienException.java
public class MoinsQueRienException extends Exception {
public MoinsQueRienException(String message) {
super(message);
}
}
throw new IllegalArgumentException("(0) . (0) *Declic is watching you*");
} else if (value < -1) {
throw new MoinsQueRienException("H4ck3r");
}
changeTickSpeed(value);
}


FormaExceptions.java
public void GameRuleHandler(String gameRuleName, String[] args) {
try {
switch (gameRuleName) {
case "randomTickSpeed":
Random
break;
case "doFi
DoFire
break;
default:
System
}
FormaExceptions.java
public void RandomTickSpeedHandler(String[] args) throws IllegalArgumentException, MoinsQueRienException {
if (args == null || args.length == 0) {
throw new IllegalArgumentException("Y a pa dé pano !");
}
int value = Integer.parseInt(args[0]);
if (value == -1) {
throw new IllegalArgumentException("(0) . (0) *Declic is watching you*");
} else if (value < -1) {
throw new MoinsQueRienException("H4ck3r");
}
changeTickSpeed(value);
}



```
                RandomTickSpeedHandler(args);
                break;

            case "doFireTick":
                DoFireTickHandler(args);
                break;

            default:
                System.err.println("PTDR t ki ?");
        }

    } catch (IllegalArgumentException | IllegalStateException e) {
        System.err.println(e.getMessage() + e.getStackTrace());

    } catch (MoinsQueRienException e) {
        crash( msg: "error 638: " + e.getMessage());

    } finally {
        System.out.println("Travail terminé...");
    }
}
```


LES STRUCTURES
DE DONNEES

FormaStruct.java

```java
// ARRAY
Item[] chest = new Item[27];
chest[0] = new Item( name: "Diamond");
Item[] enderChest = {new Item( name: "Diamond"), null /*, ...*/ };
```

FormaStruct.java

```java
// SET
Set<String> biomesVisited = new HashSet<>();
biomesVisited.add("Desert");
biomesVisited.add("Desert"); // ignoré
```

FormaStruct.java

```java
// LIST
List<Item> inventory = new ArrayList<>();
inventory.add(new Item( name: "Wood"));
inventory.removeFirst(); //.remove(0); fonctionne aussi
```

FormaStruct.java

```java
// MAP
Map<Position, Block> world = new HashMap<>();
world.put(new Position( x: 0,  y: 64,  z: 0), Block.STONE);
Map<Integer, Player> leaderboard = new TreeMap<>();
```

FormaStruct.java

```java
// STACK
Deque<Item> stack = new ArrayDeque<>();
stack.push(new Item( name: "Stick"));
stack.pop();
```

FormaStruct.java

```java
// QUEUE
Queue<Item> furnace = new ArrayDeque<>();
furnace.add(new Item( name: "Iron Ore"));
furnace.poll();
```