

World of Robots

Master 1 Génie Physiologique et Informatique

Del 6-3: Final synthesis



Group 14 : Laurène DONG, Marion DOSPITAL, Lauranne LAIR, Thibault NIGGEL, Florian PAILLAS, Ophélie SOCHARD



```

classDiagram
    class Colour {
        - Colour : enum
    }
    class CanvasRobot {
        canvas : Canvas
        randomGenerator : Random
        canvasRobotList : ArrayList<CanvasRobot>
        gameOver : boolean
        redrawable : boolean
        - x : int
        - y : int
        - n : int
        - lo : int
        - la : int
        - plo : int
        - pla : int
        - tlo : int
        - tla : int
        - qia : int
        - milieu : int
        - couleurHead : Colour
        - couleurBody : Colour
        - couleurLeg : Colour
        - couleurEye : Colour
        - corps : int
        - brasG : int
        - brasD : int
        - jambeG : int
        - jambeD : int
        - tete : int
        - oeilD : int
        - oeilG : int
        + CanvasRobot(String)
        + drawRobot(int, int): void
        + setColourBody(String): void
        + getColourBody(): String
        + valueOf(String): Colour
        + collision(): boolean
        + outOfCanvas(): boolean
        + gameOver(): void
        + ecrisVertical(int, int, String): void
        + ecrisHorizontal(int, int, String): void
    }
    class Canvas {
        - canvasSingleton : Canvas
        - frame : JFrame
        - canvas : CanvasPane
        - graphic : Graphics2D
        - backgroundColor : Color
        - canvasImage : Image
        - objects : List<Object>
        - shapes : HashMap<Object, ShapeDescription>
        - shape : Shape
        - colour : Colour
        + getCanvas(): Canvas
        + Canvas(String, int, int, Color)
        + setVisible(boolean): void
        + draw(Object, Colour, Shape): void
        + erase(Object): void
        + setForegroundColour(Colour): void
        + wait(int): void
        + redraw(): void
        + erase(): void
        + gameOver(): void
        + draw(Graphics2D): void
        + paint(Graphics): void
        + ShapeDescription(Shape, Colour)
    }
    class WorldOfRobot {
        - wordList : ArrayList<Robot>
        - canvasRobot : CanvasRobot
        - x : int
        - y : int
        - numberOfRobots : int
        + WorldOfRobot()
        + getList(): ArrayList<Robot>
        + getNumberOfRobots(): int
        + canItMove(int x, int y, Robot): boolean
        + moveAll(): void
        + moveAutomatic(int w): void
        + addRobot(Robot): void
    }
    class Robot {
        - MIN_NAME_LENGTH : int
        - MIN_POSITION : int
        - MAX_POSITION : int
        - numberOfUnnamedRobots : int
        - vitesse, direction, xPosition, yPosition : int
        - name : String
        - canvasRobot : CanvasRobot
        - canvas : Canvas
        - worldR : WorldOfRobot
        - colourBody : String
        + Robot(String, int, int, WorldOfRobot)
        + getName(): String
        + getNumberOfUnnamedRobots(): int
        + setName(String): void
        + getXPosition(): int
        + getYPosition(): int
        + setPosition(int, int): void
        + setColourBody(String): void
        + getColourBody(): String
        + canItMove(int, int): boolean
        + move(): void
        + canItDisplay(int, int): boolean
        + showRobot(): void
    }
    class Spiral {
        - trajectory : int
        - spiraleRate : int
        - xtemp : int
        - ytemp : int
        + SpiralRobot(String, WorldOfRobot)
        + move(): void
    }
    class Couleur {
        - randomGenerator : Random
        - goingRight : boolean
        - couleur : int
        + Couleur(String, WorldOfRobot)
        + move(): void
    }
    class Savant {
        - parier : String
        - goingRight : boolean
        - goingDown : boolean
        + Savant(String, WorldOfRobot)
        + setParier(String): void
        + EffectuerCalcul(double, double, char): double
        + move(): void
    }
    class Randonne {
        + Randonne(String, int, int, WorldOfRobot)
        + Random010(): int
        + move()
    }
    class Danseur {
        + Danseur(String, WorldOfRobot)
        + move(): void
        + Valse(): void
    }
    class Dessin {
        - colourLine : String
        - lesCouleurs : String[]
        - trace : boolean
        + Dessin(String, int, int, WorldOfRobot)
        + getColourLine(): String
        + setColourLine(String): void
        + rendCouleur(String): String
        + setTracer(boolean): void
        + moveHorizontal(int): void
        + moveVertical(int): void
        + move(): void
    }
    CanvasRobot --> Canvas
    CanvasRobot --> WorldOfRobot
    CanvasRobot --> Robot
    Canvas --> WorldOfRobot
    Canvas --> Robot
    WorldOfRobot --> Robot
    Robot <|-- Danseur
    Robot <|-- Dessin
    
```

- **Inheritance**

Inheritance also allows for the protection of attributes in different classes. To call methods from the parent class, we used "super".



- **Override**

Thanks to the use of the presence of inheritance, we've been able to experiment with the use of overrides. With the Move() method for example. It is present in the parent Robot() class, but it's empty. It is also present in the Robots() daughter classes, such as Random() or Colour(). However, in these daughter classes, it is filled with different functions depending on the daughter class.

- **Javadocs**

During this project, we were able to implement the Javadocs principle. This involves adding comments to classes, constructor(s), and methods to explain how they work, as well as providing information about method parameters or class attributes. We tried to write it in a clear and concise manner so that anyone unfamiliar with the code can understand how our program functions."

- **Debugging**

To find out where our errors were coming from, we used debugging when testing our methods and unit tests. This method has allowed us to be more efficient in finding solutions.

The realization of this project was complicated and dense. Despite this, we learned to communicate effectively, organize the work in time, share our understanding and our questions to ensure that everyone follows the project and understands well. In terms of skills, we were quite complementary which was a considerable advantage and we were able to motivate ourselves in moments of relaxation. The team leader, the secretary and the time manager allowed us to carry out this project but each member of the group played a decisive role, whether in the understanding of English, concepts and programming, synthesis writing and work organization. For World of Zuul, we will seek to improve the organization, review certain concepts such as debugging and motivation throughout the project.