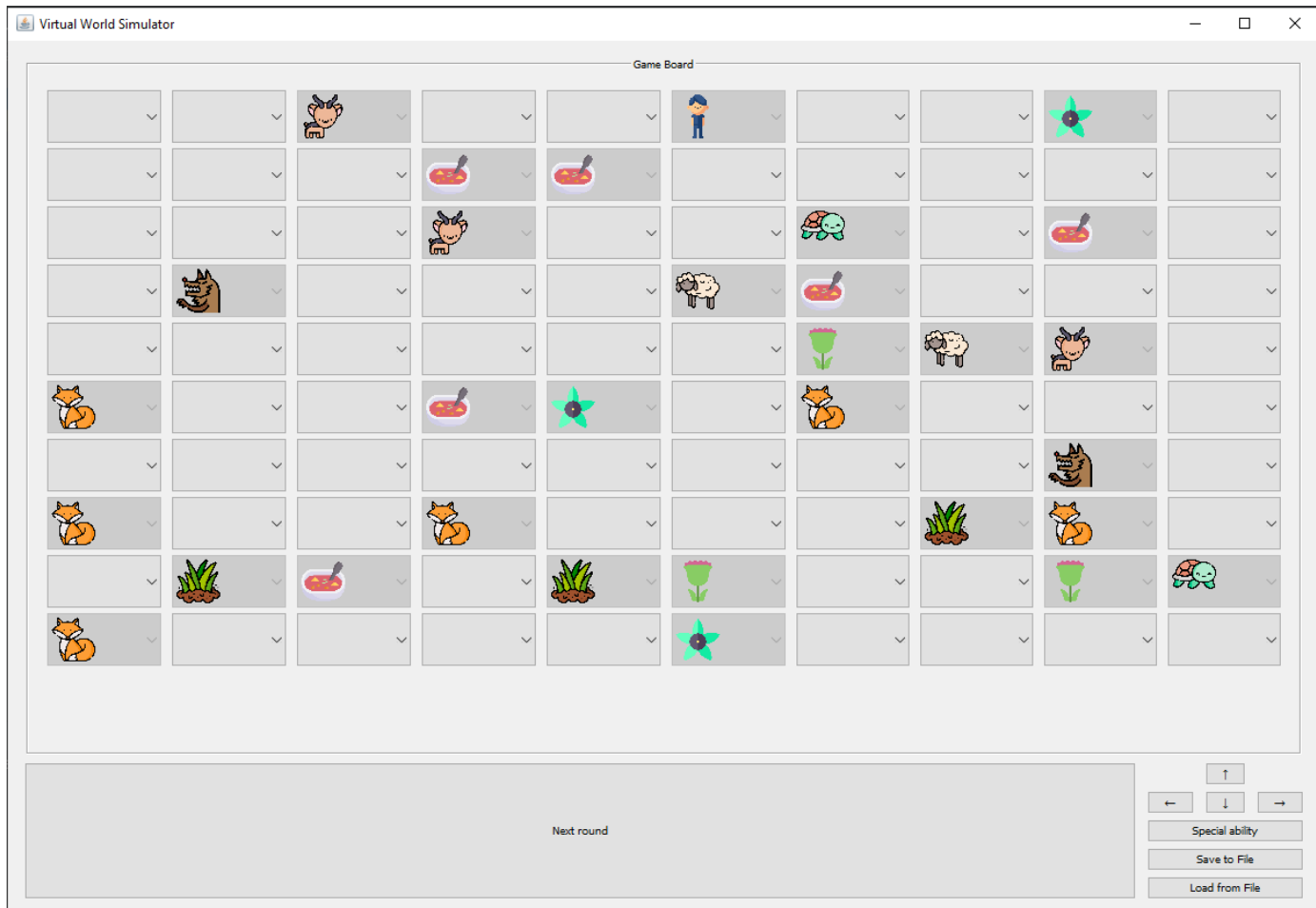
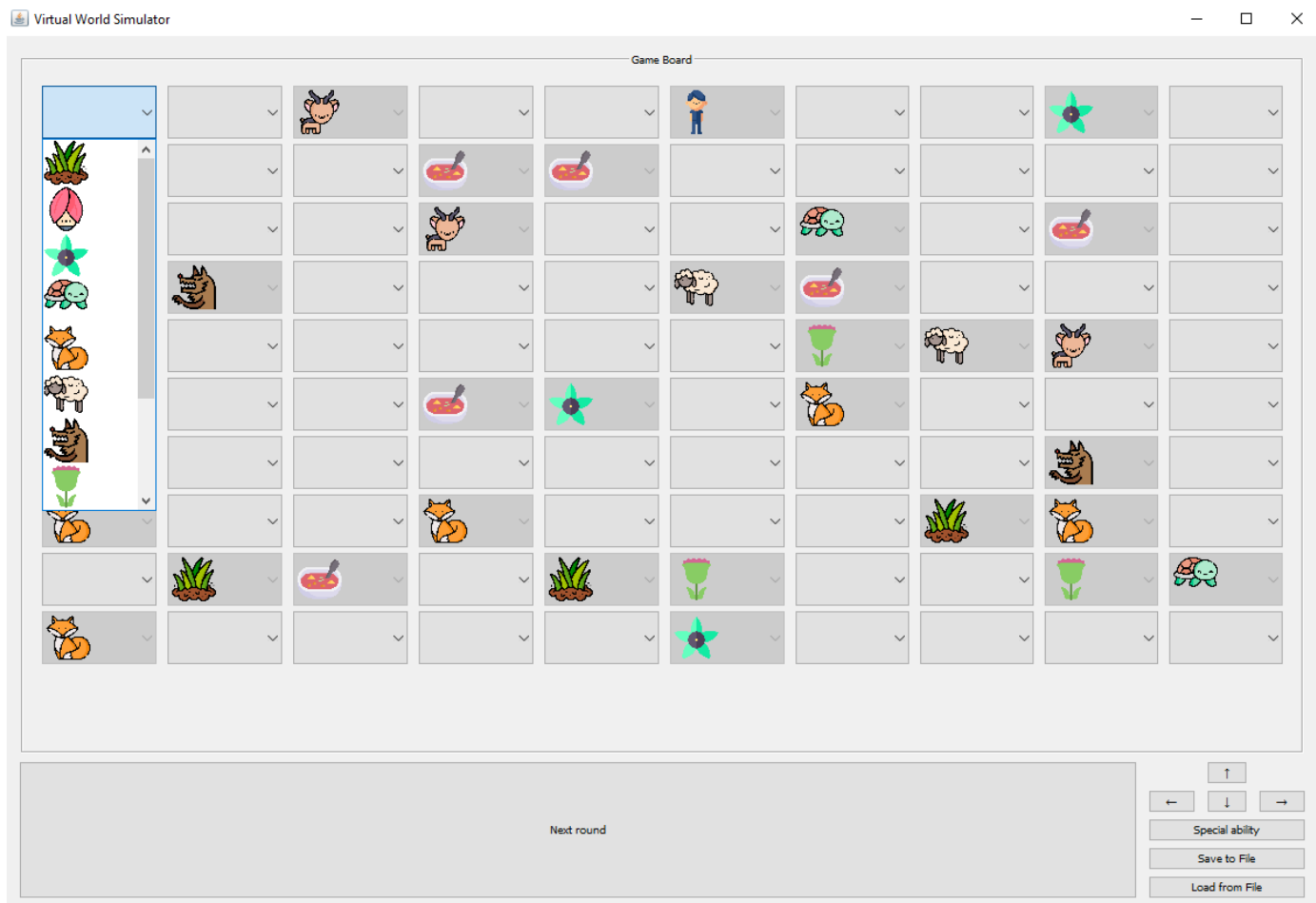


Points done 4/5.

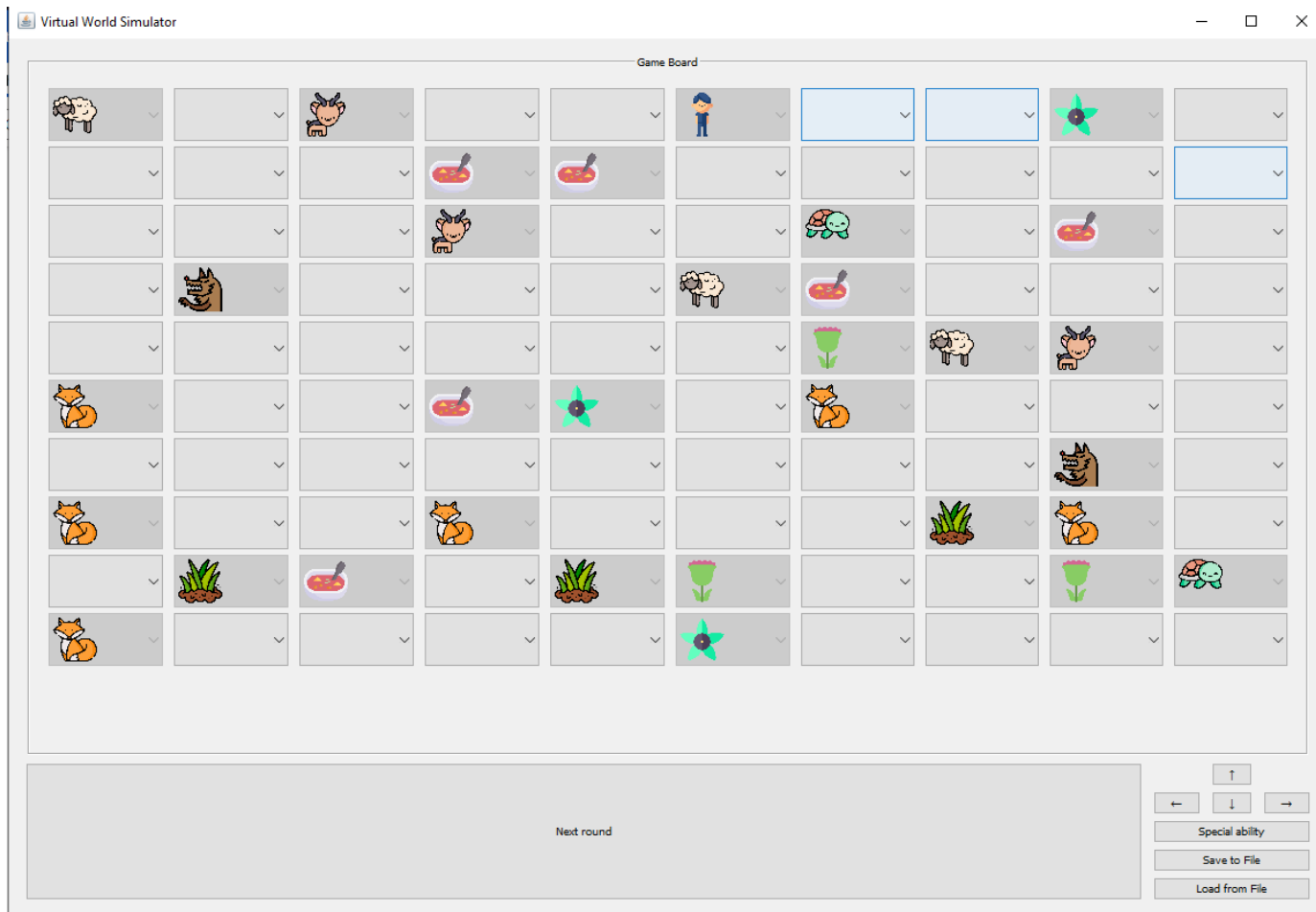
Look of the program:



Main fields of the game are comboboxes that are located on a grid 10x10. There are let's say two types of those comboboxes. First is an empty type. Which we can click on, and add an organism that we want to:

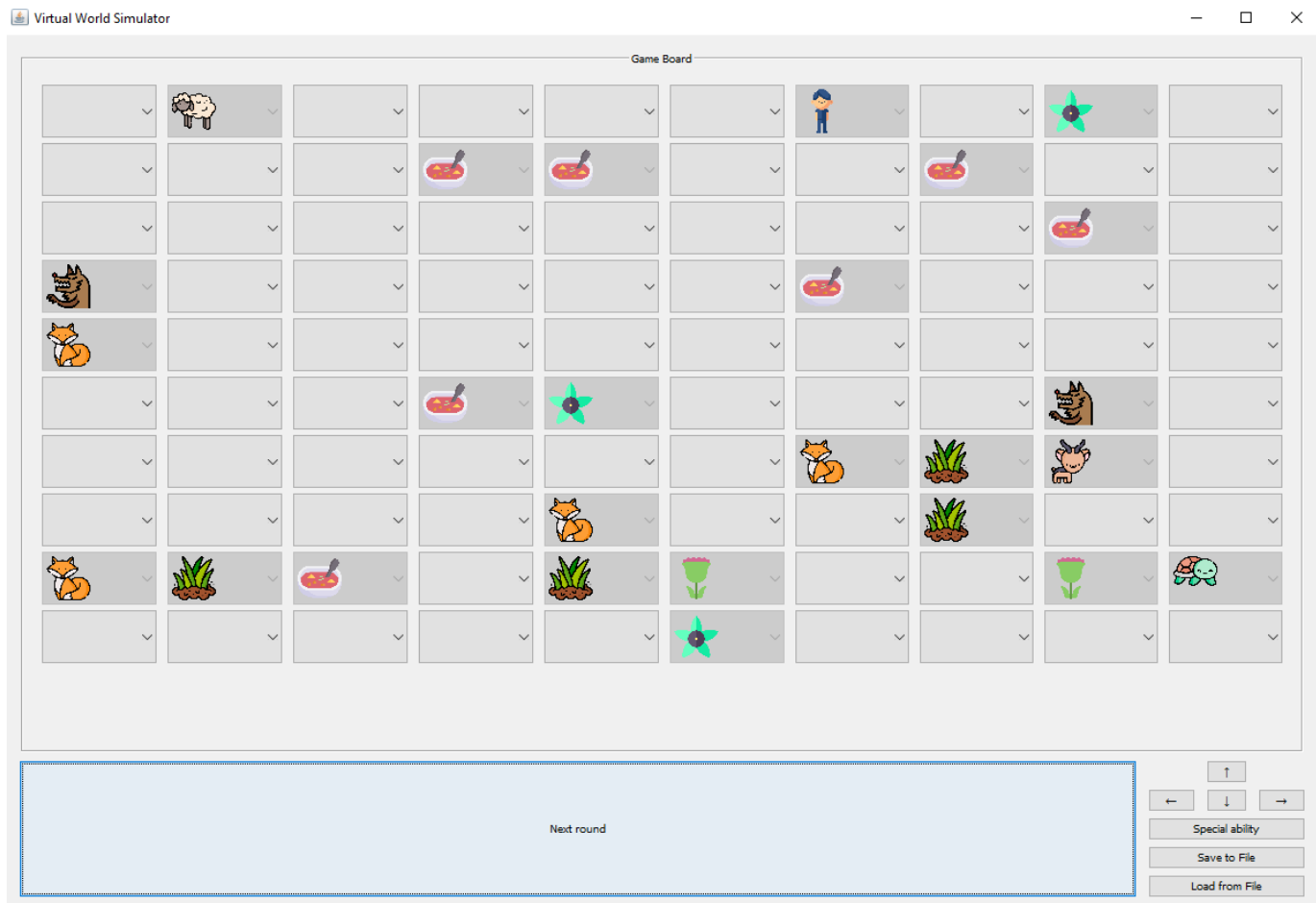


When we click on the empty field a dropdown list appears and we can choose which organism we want to add. Let's say that we choose a sheep:

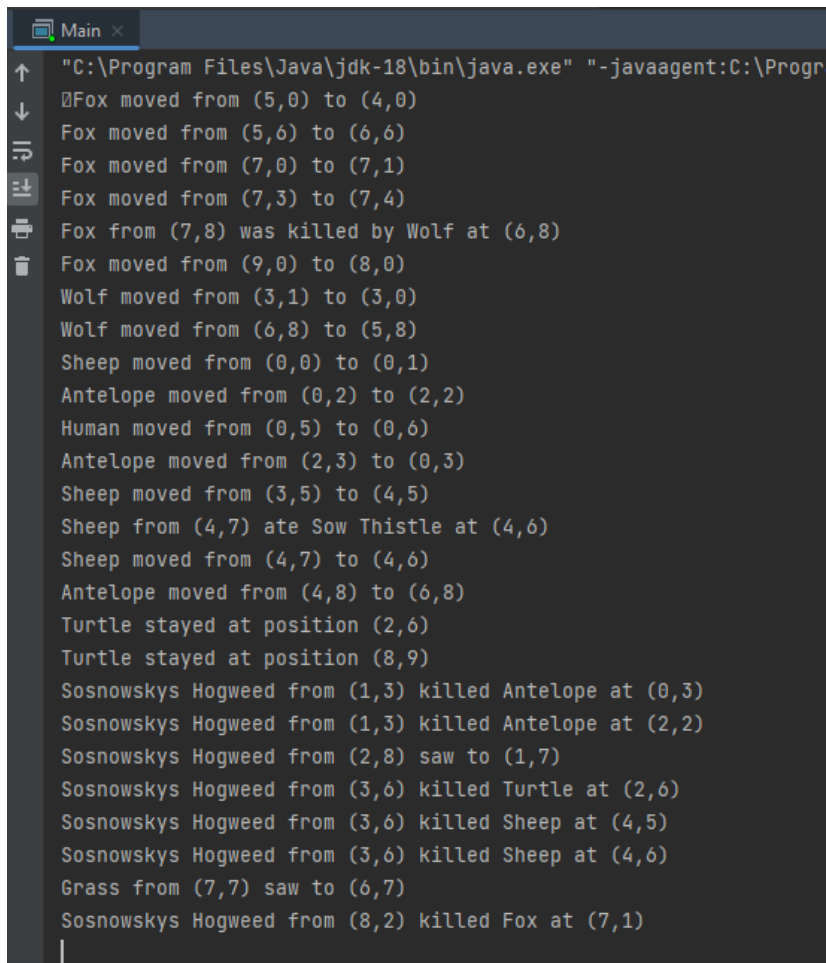


And as we can see sheep was added to the board and this field is now locked. There are also 8 other buttons on this board, big button for playing a next round, buttons for choosing the direction in which human will move, button for the special ability of a human and two buttons for saving the state of the game into a file and loading it from a file.

When we lock the position in which we want to go, we can click next round.

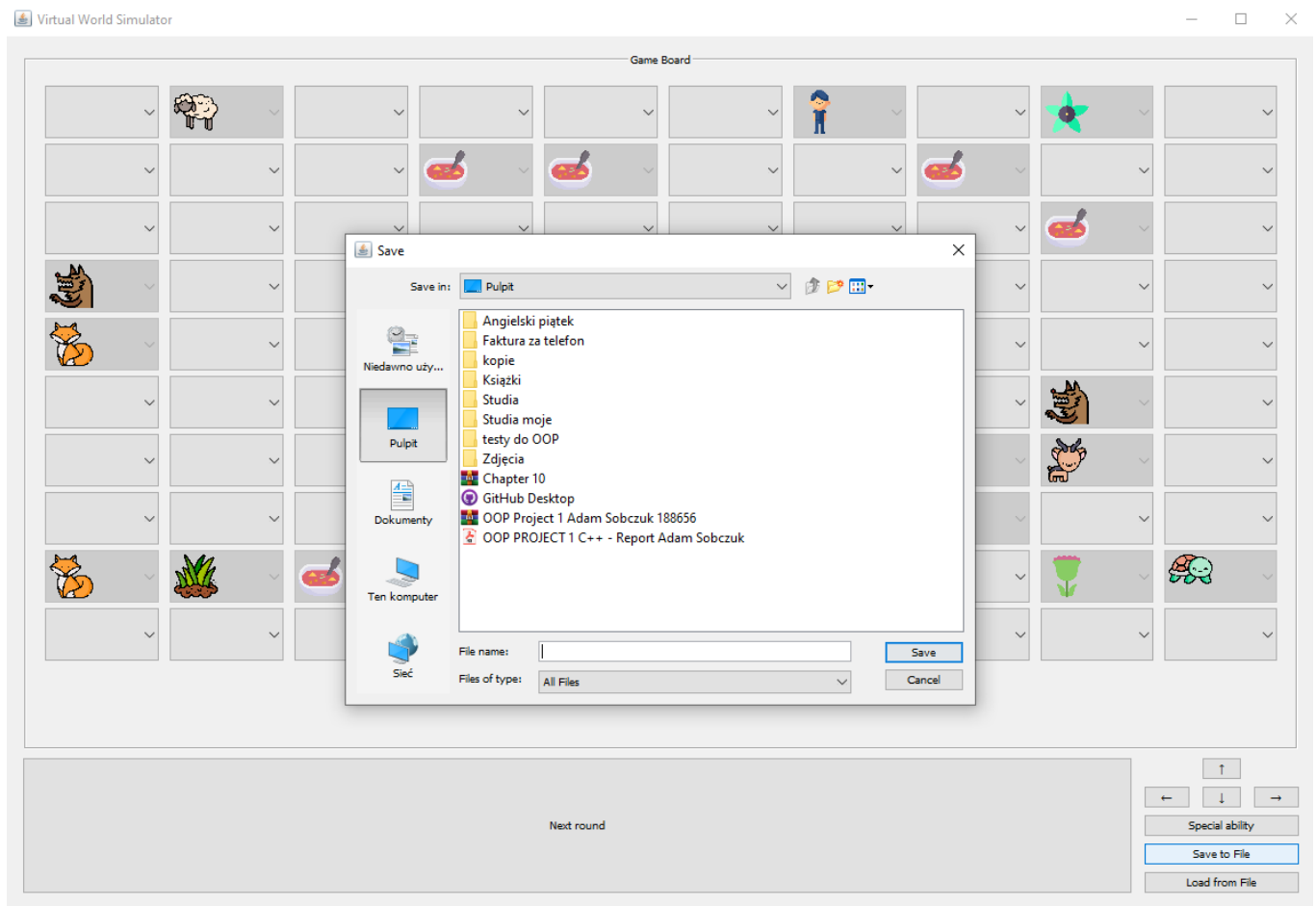


As we can see the board changed in accordance to the logic of the project. Action logs are printed in the console:

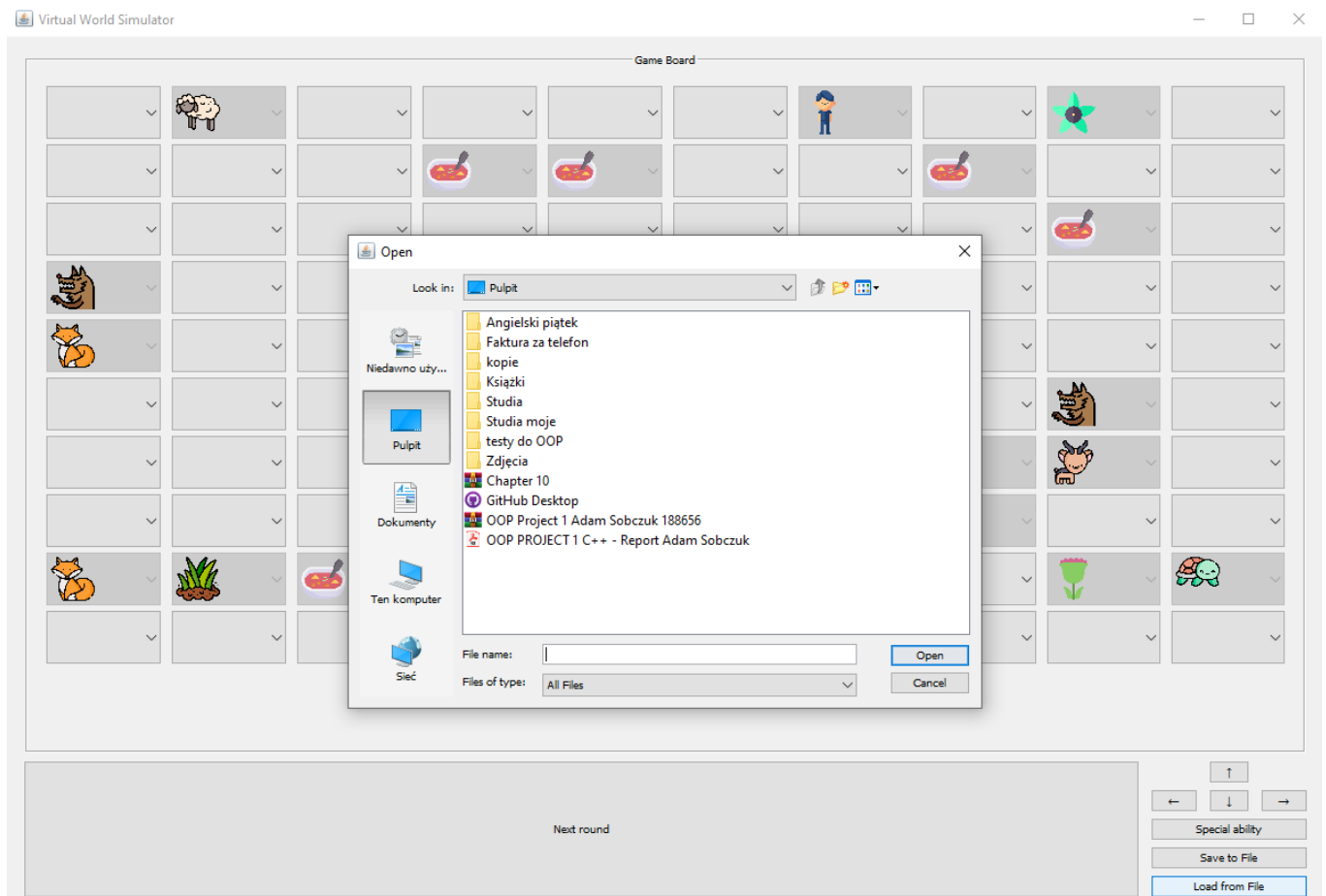
A screenshot of a Java IDE's console window. The window has a title bar with a single tab labeled 'Main'. On the left side, there is a vertical toolbar with icons for running, stepping through, and other debugging actions. The main area of the window displays a series of text logs in a monospaced font. The logs describe the movements and actions of various game entities like Fox, Wolf, Sheep, Antelope, Turtle, and Hogweed on a grid. The text is as follows:

```
"C:\Program Files\Java\jdk-18\bin\java.exe" "-javaagent:C:\Progr
Fox moved from (5,0) to (4,0)
Fox moved from (5,6) to (6,6)
Fox moved from (7,0) to (7,1)
Fox moved from (7,3) to (7,4)
Fox from (7,8) was killed by Wolf at (6,8)
Fox moved from (9,0) to (8,0)
Wolf moved from (3,1) to (3,0)
Wolf moved from (6,8) to (5,8)
Sheep moved from (0,0) to (0,1)
Antelope moved from (0,2) to (2,2)
Human moved from (0,5) to (0,6)
Antelope moved from (2,3) to (0,3)
Sheep moved from (3,5) to (4,5)
Sheep from (4,7) ate Sow Thistle at (4,6)
Sheep moved from (4,7) to (4,6)
Antelope moved from (4,8) to (6,8)
Turtle stayed at position (2,6)
Turtle stayed at position (8,9)
Sosnowskys Hogweed from (1,3) killed Antelope at (0,3)
Sosnowskys Hogweed from (1,3) killed Antelope at (2,2)
Sosnowskys Hogweed from (2,8) saw to (1,7)
Sosnowskys Hogweed from (3,6) killed Turtle at (2,6)
Sosnowskys Hogweed from (3,6) killed Sheep at (4,5)
Sosnowskys Hogweed from (3,6) killed Sheep at (4,6)
Grass from (7,7) saw to (6,7)
Sosnowskys Hogweed from (8,2) killed Fox at (7,1)
```

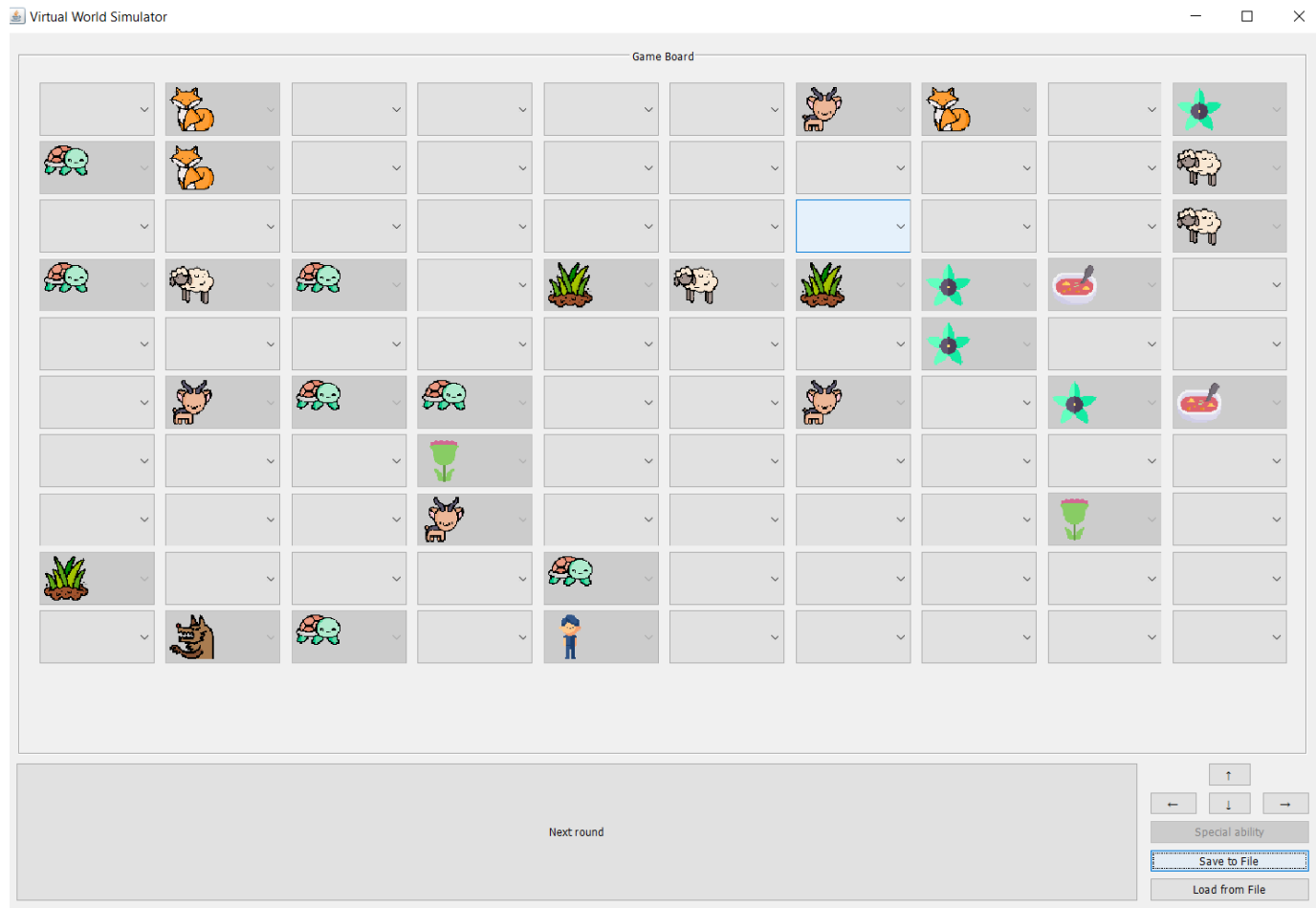
Button save to file prompts this window in which we can choose and name a file in which we want to save the state of the game.



Similarly with the load from file button:

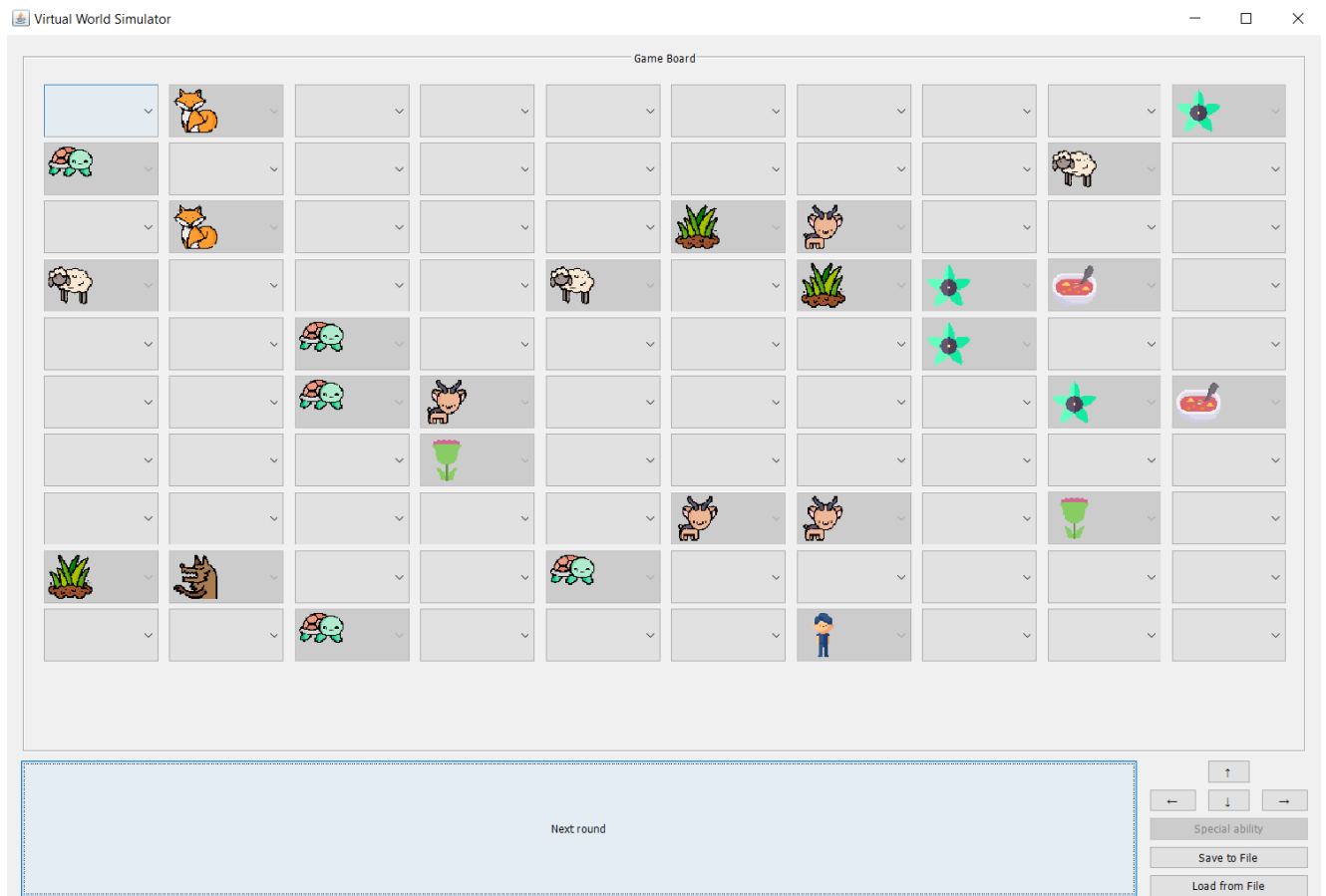


The last feature would be the special ability button, when we click it the human's antelope speed ability activates for the 5 next rounds. And then it desactivates and leaves the button not possible to enable. However, in our case I had to try it in a different game, because my human was unfortunately killed by sosnowsky's hogweed.



Now special ability is on.





And as we can see human moved two fields.

In this project I divided the visualization part in two. Firstly, there's view which contains form and two classes `jcomboboxanimal` and `jcomboboxanimalrenderer`. In main form all comboboxes are created and added to the board.

1 usage

```
public MainForm() {  
    frame = new JFrame( title: "Virtual World Simulator");  
    frame.setContentPane(mainPanel);  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    frame.pack();  
}  
  
public void createUIComponents() {  
    jComboBoxAnimalList = new ArrayList<>();  
    comboBoxAnimal00 = new JComboBoxAnimal<>( xCord: 0, yCord: 0, organism: null);  
    comboBoxAnimal01 = new JComboBoxAnimal<>( xCord: 0, yCord: 1, organism: null);  
    comboBoxAnimal02 = new JComboBoxAnimal<>( xCord: 0, yCord: 2, organism: null);  
    comboBoxAnimal03 = new JComboBoxAnimal<>( xCord: 0, yCord: 3, organism: null);  
    comboBoxAnimal04 = new JComboBoxAnimal<>( xCord: 0, yCord: 4, organism: null);  
    comboBoxAnimal05 = new JComboBoxAnimal<>( xCord: 0, yCord: 5, organism: null);  
    comboBoxAnimal06 = new JComboBoxAnimal<>( xCord: 0, yCord: 6, organism: null);  
    comboBoxAnimal07 = new JComboBoxAnimal<>( xCord: 0, yCord: 7, organism: null);  
    comboBoxAnimal08 = new JComboBoxAnimal<>( xCord: 0, yCord: 8, organism: null);  
    comboBoxAnimal09 = new JComboBoxAnimal<>( xCord: 0, yCord: 9, organism: null);
```

```
    jComboBoxAnimalList.add(comboBoxAnimal00);  
    jComboBoxAnimalList.add(comboBoxAnimal01);  
    jComboBoxAnimalList.add(comboBoxAnimal02);  
    jComboBoxAnimalList.add(comboBoxAnimal03);  
    jComboBoxAnimalList.add(comboBoxAnimal04);  
    jComboBoxAnimalList.add(comboBoxAnimal05);  
    jComboBoxAnimalList.add(comboBoxAnimal06);  
    jComboBoxAnimalList.add(comboBoxAnimal07);  
    jComboBoxAnimalList.add(comboBoxAnimal08);  
    jComboBoxAnimalList.add(comboBoxAnimal09);
```

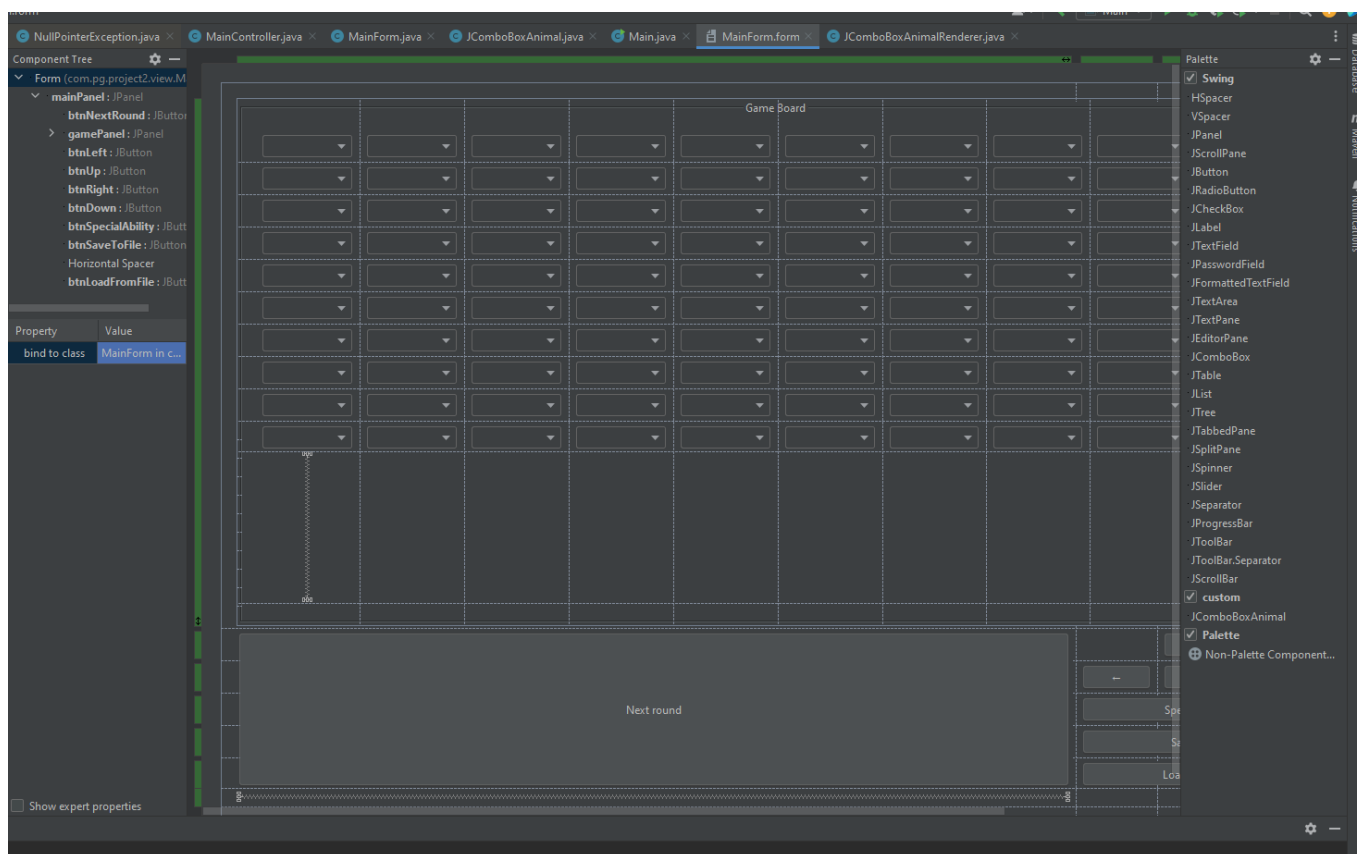
Representations of animals are added to the iconmap in jcomboboxanimalrenderer

```

public JComboBoxAnimalRenderer() {
    iconMap.put(OrganismRepr.ANTELOPE, getImageIcon( name: "antelope.png"));
    iconMap.put(OrganismRepr.BELLADONNA, getImageIcon( name: "belladonna.png"));
    iconMap.put(OrganismRepr.SOSNOWSKYS_HOGWEED, getImageIcon( name: "borscht.png"));
    iconMap.put(OrganismRepr.SOW_THISTLE, getImageIcon( name: "flower.png"));
    iconMap.put(OrganismRepr.GRASS, getImageIcon( name: "grass.png"));
    iconMap.put(OrganismRepr.GUARANA, getImageIcon( name: "guarana.png"));
    iconMap.put(OrganismRepr.HUMAN, getImageIcon( name: "man.png"));
    iconMap.put(OrganismRepr.SHEEP, getImageIcon( name: "sheep.png"));
    iconMap.put(OrganismRepr.TURTLE, getImageIcon( name: "turtle.png"));
    iconMap.put(OrganismRepr.FOX, getImageIcon( name: "fox.png"));
    iconMap.put(OrganismRepr.WOLF, getImageIcon( name: "wolf.png"));
    iconMap.put(OrganismRepr.EMPTY, null);
}

```

For the look of our game I also used swing UI designer framework.



The second part is the controller which links our view with the model.

1 usage

```
public MainController() {  
    mainForm = new MainForm();  
    world = new World();  
}
```

1 usage

```
public void control() {  
    init();  
    addActionListenersForComboBoxes();  
    mainForm.getBtnLeft().addActionListener(e -> onClickBtnHumanMove(e));  
    mainForm.getBtnDown().addActionListener(e -> onClickBtnHumanMove(e));  
    mainForm.getBtnRight().addActionListener(e -> onClickBtnHumanMove(e));  
    mainForm.getBtnUp().addActionListener(e -> onClickBtnHumanMove(e));  
    mainForm.getBtnSpecialAbility().addActionListener(e -> onClickBtnSpecialAbility());  
    mainForm.getBtnSaveToFile().addActionListener(e -> onClickBtnSaveToFile());  
    mainForm.getBtnLoadFromFile().addActionListener(e -> onClickBtnLoadFromFile());  
    mainForm.getBtnNextRound().addActionListener(e -> onClickBtnNextRound());  
}
```

1 usage

```
private void addActionListenersForComboBoxes() {  
    mainForm.getJComboBoxAnimalList().forEach(c -> c.addActionListener(this::onChangeComboBox));  
}
```

1 usage

```
private void init() {  
    mainForm.getFrame().setVisible(true);  
    performWorldInit();  
}
```

3 usages

```
private void reloadComboBoxes() {  
    for (int x = 0; x < 10; x++)  
        for (int y = 0; y < 10; y++)  
            mainForm.setOrganism(x, y, world.getWorld().get(x).get(y));  
}
```

1 usage

```
private void performWorldInit() {  
    world.initializePopulation();  
    reloadComboBoxes();  
}
```

1 usage

```
private void performPlace(int x, int y, OrganismRepr selectedOrganismRepr, JComboBoxAnimal jComboBoxAnimal) {  
    Organism organism = selectedOrganismRepr.getOrg();  
    if (!Objects.isNull(organism)) {  
        organism.setWorld(world).setX(x).setY(y).setSymbol(selectedOrganismRepr);  
        organism.setAge(0);  
    }  
    jComboBoxAnimal.setOrganism(organism);  
    world.place(x, y, organism);  
}
```

1 usage

```
private void onChangeComboBox(ActionEvent e) {  
    JComboBoxAnimal<?> jComboBoxAnimal = ((JComboBoxAnimal<?>) e.getSource());  
    OrganismRepr selectedOrganismRepr = ((OrganismRepr) Objects.requireNonNull(jComboBoxAnimal.getSelectedItem()));  
    if (Objects.isNull(world.getWorld().get(jComboBoxAnimal.getXCord()).get(jComboBoxAnimal.getYCord())))  
        performPlace(jComboBoxAnimal.getXCord(), jComboBoxAnimal.getYCord(), selectedOrganismRepr, jComboBoxAnimal);  
}
```

```

4 usages
private void onClickBtnHumanMove(ActionEvent e) {
    humanDirection = MoveRepr.findByStrRepr(((JButton) e.getSource()).getText()).getIntRepr();
}

1 usage
private void onClickBtnSpecialAbility() {
    world.setRoundTurn(1);
    mainForm.getBtnSpecialAbility().setEnabled(false);
}

1 usage
private void onClickBtnSaveToFile() {
    try {
        world.saveWorld(getFile( saveMode: true));
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}

1 usage
private void onClickBtnLoadFromFile() {
    try {
        world.readWorld(getFile( saveMode: false));
        reloadComboBoxes();
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }
}

1 usage
private void onClickBtnNextRound() {
    world.makeTurn(humanDirection);
    reloadComboBoxes();
    mainForm.getBtnSpecialAbility().setEnabled(world.getRoundTurn() == 0);
}

```

Method control contains adding action listeners for all combo boxes and for all the buttons as well. It also calls the method init which makes our panel visible and which calls another method performworld init which calls method initialize population from the world class. More about that later. Another method would be reload comboboxes that updates our panel after a next round or after adding an organism. Method performPlace is used for adding the organism to our model after choosing it from a list. And methods that call methods from world upon clicking buttons.

Now it's time for the model:

I'll skip the parts from the model that are very similar to those in C++. Logic stays the same. And I'll show only the differences just for clarity.

```

public enum OrganismRepr {
    4 usages
    GRASS( repr: "|", new Grass( world: null)), GUARANA( repr: "g", new Guarana( world: null)),
    4 usages
    BELLADONNA( repr: "b", new Belladonna( world: null)), TURTLE( repr: "t", new Turtle( world: null)),
    4 usages
    FOX( repr: "f", new Fox( world: null)), SHEEP( repr: "s", new Sheep( world: null)),
    4 usages
    WOLF( repr: "w", new Wolf( world: null)), SOW_THISTLE( repr: "m", new SowThistle( world: null)),
    4 usages
    SOSNOWSKYS_HOGWEED( repr: "H", new SosnowskysHogweed( world: null)), HUMAN( repr: "C", new Human( world: null)),
    4 usages
    ANTELOPE( repr: "a", new Antelope( world: null)), EMPTY( repr: " ", org: null);

    private String repr;
    private Organism org;

    public OrganismRepr[] getValues() { return values(); }

    1 usage
    public static OrganismRepr findByCode(String code) {
        return Arrays.stream(values()).filter(e -> e.getRepr().equals(code)).findFirst().orElse(EMPTY);
    }

    1 usage
    public static OrganismRepr[] valuesForChoose() {
        return new OrganismRepr[]{GRASS, GUARANA, BELLADONNA, TURTLE, FOX, SHEEP, WOLF, SOW_THISTLE, SOSNOWSKYS_HOGWEED,
            HUMAN, ANTELOPE, EMPTY
        };
    }
}

```

There's a new enum class which is used in adding new organisms to the board.

```

public enum OrganismRepr {
    4 usages
    GRASS( repr: "|", new Grass( world: null)), GUARANA( repr: "g", new Guarana( world: null)),
    4 usages
    BELLADONNA( repr: "b", new Belladonna( world: null)), TURTLE( repr: "t", new Turtle( world: null)),
    4 usages
    FOX( repr: "f", new Fox( world: null)), SHEEP( repr: "s", new Sheep( world: null)),
    4 usages
    WOLF( repr: "w", new Wolf( world: null)), SOW_THISTLE( repr: "m", new SowThistle( world: null)),
    4 usages
    SOSNOWSKYS_HOGWEED( repr: "H", new SosnowskysHogweed( world: null)), HUMAN( repr: "C", new Human( world: null)),
    4 usages
    ANTELOPE( repr: "a", new Antelope( world: null)), EMPTY( repr: " ", org: null);

    private String repr;
    private Organism org;

    public OrganismRepr[] getValues() { return values(); }

    1 usage
    public static OrganismRepr findByCode(String code) {
        return Arrays.stream(values()).filter(e -> e.getRepr().equals(code)).findFirst().orElse(EMPTY);
    }

    1 usage
    public static OrganismRepr[] valuesForChoose() {
        return new OrganismRepr[]{GRASS, GUARANA, BELLADONNA, TURTLE, FOX, SHEEP, WOLF, SOW_THISTLE, SOSNOWSKYS_HOGWEED,
            HUMAN, ANTELOPE, EMPTY
        };
    }
}

```

```

public class Pair<L, R> {
    1 usage
    private L left;
    1 usage
    private R right;

    1 usage
    public void set(L left, R right) {
        this.left = left;
        this.right = right;
    }
}

```

I use class pair for calculating coordinates for organisms to move and to breed and whatnot.



```

public enum MoveRepr {
    LEFT( strRepr: "←", intRepr: 3), DOWN( strRepr: "↓", intRepr: 2), RIGHT( strRepr: "→", intRepr: 1), UP( strRepr: "↑", intRepr: 0), UNKNOWN( strRepr: "", intRepr: -1);

    private String strRepr;
    private int intRepr;

    public MoveRepr[] getValues() { return values(); }

    1 usage
    public static MoveRepr findByStrRepr(String strRepr) {
        return Arrays.stream(values()).filter(e -> e.getStrRepr().equals(strRepr)).findFirst().orElse(UNKNOWN);
    }
}

```

Another enum class used for choosing direction in which human moves.

```

1 usage
public void saveWorld(File file) throws IOException {
    FileOutputStream fout = new FileOutputStream(file);
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fout));
    for (int i = 0; i < sizeX; i++)
        for (int j = 0; j < sizeY; j++)
            if (!Objects.isNull(world.get(i).get(j))) {
                bw.write( str world.get(i).get(j).getSymbol().getRepr() + " "
                    + world.get(i).get(j).getX() + " "
                    + world.get(i).get(j).getY() + " "
                    + world.get(i).get(j).getStrength() + " "
                    + world.get(i).get(j).getAge() + " ");
                bw.newLine();
            }
    bw.close();
}

1 usage
public World readWorld(File file) throws IOException {
    initWorld();
    FileInputStream fin = new FileInputStream(file);
    BufferedReader br = new BufferedReader(new InputStreamReader(fin));
    String line = br.readLine();
    while (!Objects.isNull(line)) {
        String[] elements = line.split( regex: " ");
        Organism org = this.chooseOrganism(OrganismRepr.findByCode(elements[0]));
        org.setStrength(Integer.valueOf(elements[3]));
        org.setAge(Integer.valueOf(elements[4]));
        this.place(Integer.parseInt(elements[1]), Integer.parseInt(elements[2]), org);
        line = br.readLine();
    }
    br.close();
    return this;
}

```

Methods saveworld and readworld look much neater in java 😊

Pretty much everything else looks very similar to C++, the only differences are the type used e.g. class Integer instead of int or the world board being a list of lists instead of vector of vectors. Another example would be that sorting ages and initiatives is with sets and lists not with sets and vectors.

And I think that would be pretty much all.

Thank you for your attention.