Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Новосибирский государственный технический университет»

Кафедра вычислительной техники

**Отчет**

По лабораторной работе №4

По дисциплине: «Технологии и методы программирования»

Многопотоковые приложения

Вариант 8

Выполнили: Павлюк А.С. Преподаватель: Копылова О.А.

Казанцев К.О.

Группа: АВТ-008

Факультет: АВТ

Новосибирск, 2022

**Цель работы:**

Изучить особенности реализации и работы потоков в Java, управлением приоритетами потоков и синхронизацией потоков.

**Задание к лабораторной работе:**

Доработать программу, созданную в лабораторной работу № 3:

1. создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;
2. реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;
3. синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;
4. добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов. Реализовать через засыпание/пробуждение потоков (методы wait() и notify());
5. добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

*Вариант 8*

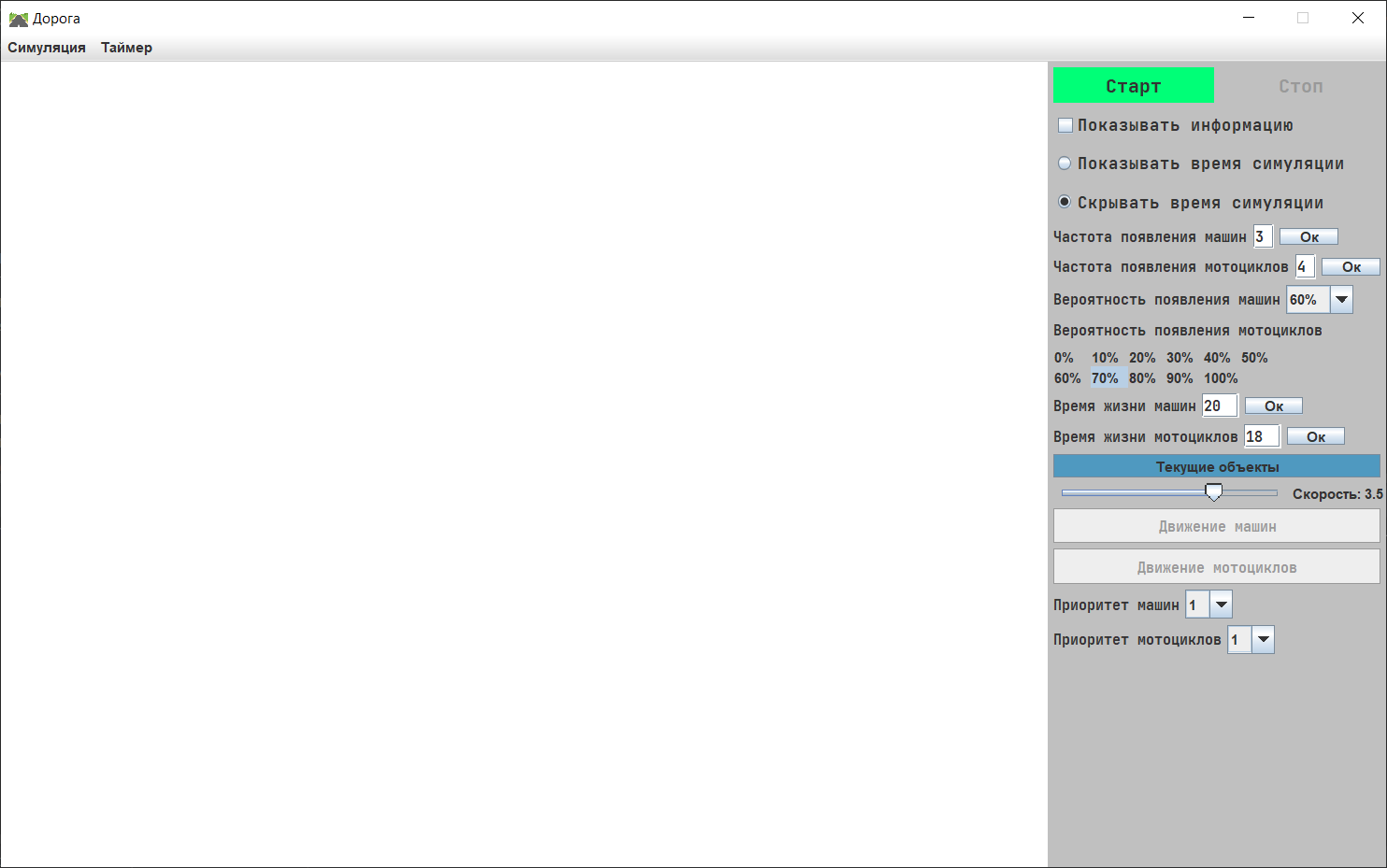
1.Автомобили двигаются по оси X от одного края области симуляции до другого со скоростью V.

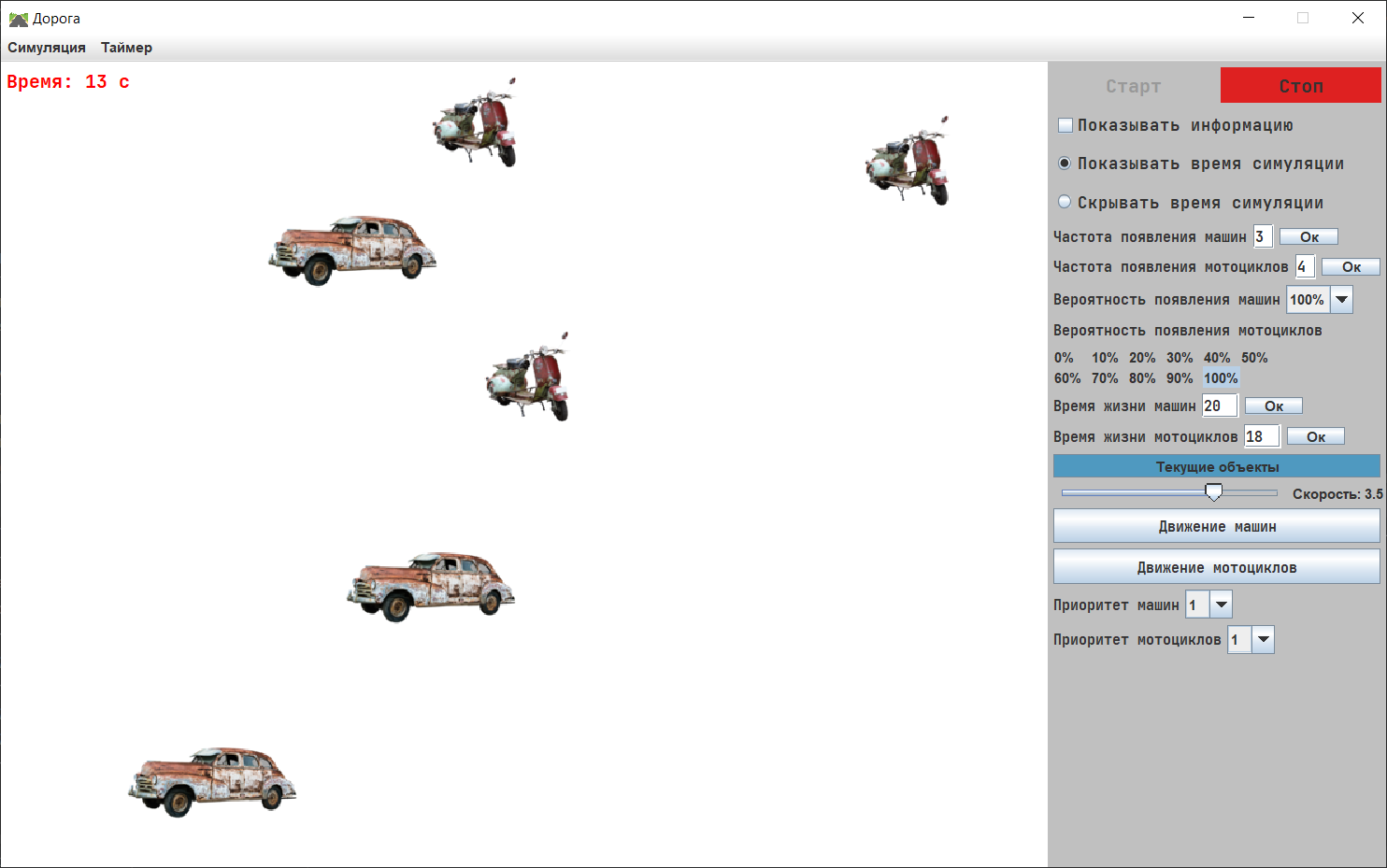
2.Мотоциклы двигаются по оси Y от одного края области симуляции до другого со скоростью V.

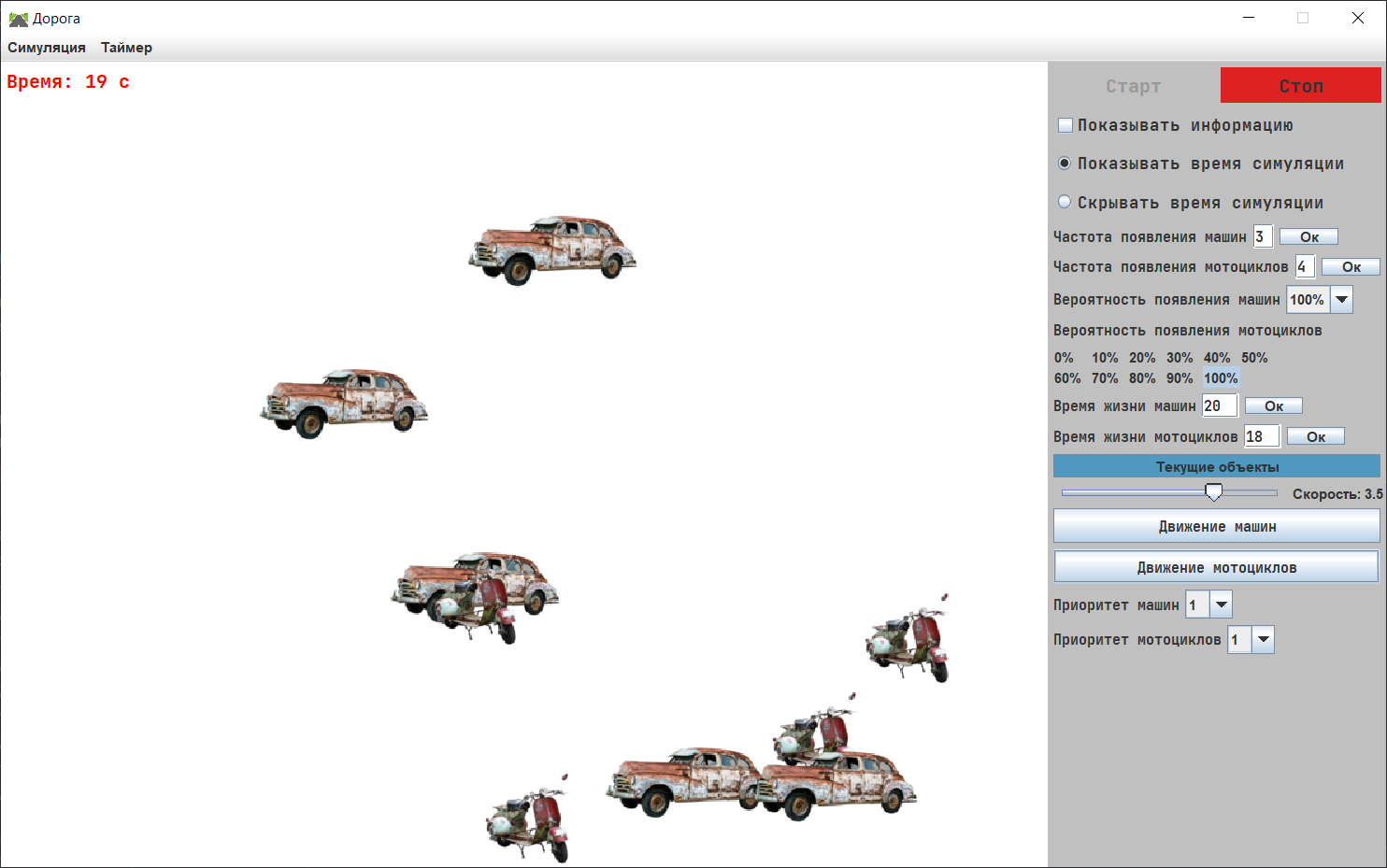
**Выполнение работы:**

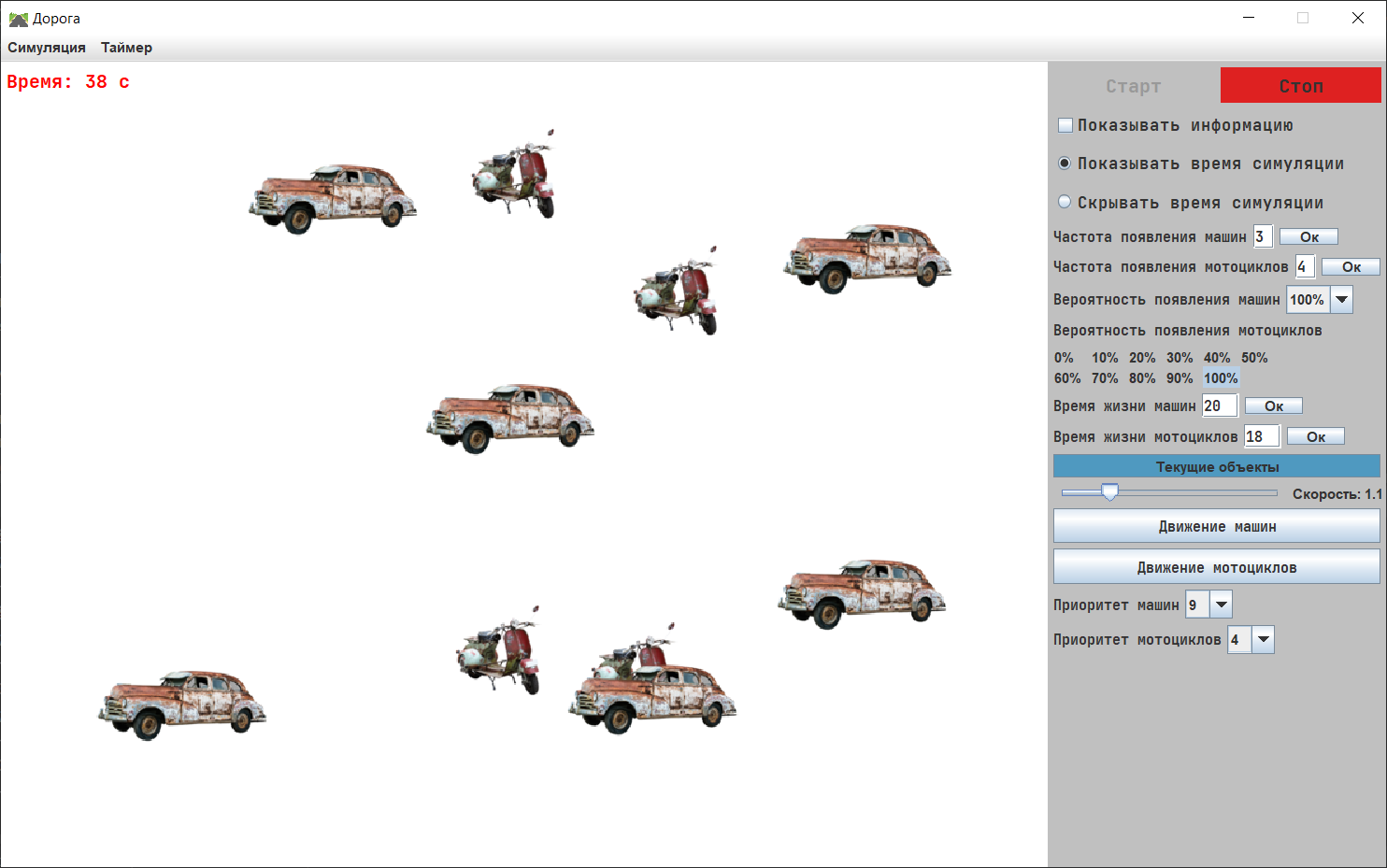
Был создан абстрактный класс BaseAI, содержащий состояние потоков. Два наследника CarAI и MotorbikeAI реализуют поведение потоков. В интерфейс IBehaviour добавлен метод move(), в Car и Motorbike они определяют логику поведения объектов на экране. Пользователь теперь может по нажатию кнопок запускать/останавливать движение объектов, изменять их скорость и устанавливать приоритеты потоков.

**Пример выполнения работы**









**Вывод:** в ходе выполнения лабораторной работы были изучены особенности реализации и работы потоков в Java, управление приоритетами потоков и синхронизация потоков.

**Класс BaseAI:**

**package** nstu;  
  
**public abstract class** BaseAI **extends** Thread {  
 **public boolean isMoving** = **true**;  
}

**Класс CarAI:**

**package** nstu.vehicles;  
  
**import** nstu.BaseAI;  
**import** nstu.MyFrame;  
  
**import static** nstu.Habitat.\*;  
  
**public class** CarAI **extends** BaseAI {  
 **public final** Object **carMonitor**;  
  
 **public** CarAI() {  
 **carMonitor** = **new** Object();  
 setPriority(1);  
 }  
  
 @Override  
 **public void** run() {  
 **while** (**isMoving**) {  
 **synchronized** (**carMonitor**) {  
 **if** (!MyFrame.*startCarMoving*) {  
 **try** {  
 **carMonitor**.wait();  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Поток прерван: "** + e.getMessage());  
 }  
 }  
  
 **for** (**int** i = 0; i < ***vehicles***.size(); i++) {  
 Vehicle v = ***vehicles***.get(i);  
 **if** (v **instanceof** Car) {  
 ((Car) v).move();  
 }  
 }  
 }  
 **try** {  
 Thread.*sleep*(10);  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Поток прерван: "** + e.getMessage());  
 }  
 }  
 }  
}

**Класс MotorbikeAI:**

**package** nstu.vehicles;  
  
**import** nstu.BaseAI;  
**import** nstu.MyFrame;  
  
**import static** nstu.Habitat.***vehicles***;  
  
**public class** MotorbikeAI **extends** BaseAI {  
 **public final** Object **motoMonitor**;  
  
 **public** MotorbikeAI() {  
 **motoMonitor** = **new** Object();  
 setPriority(1);  
 }  
  
 @Override  
 **public void** run() {  
 **while** (**isMoving**) {  
 **synchronized** (**motoMonitor**) {  
 **if** (!MyFrame.*startMotoMoving*) {  
 **try** {  
 **motoMonitor**.wait();  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Поток прерван: "** + e.getMessage());  
 }  
 }  
  
 **for** (**int** i = 0; i < ***vehicles***.size(); i++) {  
 Vehicle v = ***vehicles***.get(i);  
 **if** (v **instanceof** Motorbike) {  
 ((Motorbike) v).move();  
 }  
 }  
 }  
 **try** {  
 Thread.*sleep*(10);  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Поток прерван: "** + e.getMessage());  
 }  
 }  
 }  
}

**Изменения в интерфейсе IBehaviour:**

@FunctionalInterface  
**public interface** IBehaviour {  
 **void** move();  
}

**Изменения в классе Habitat:**

**public static float** *speed* = 3.5f;

**Изменения в классе Vehicle:**

**protected float x**, **y**, **speed**;

**Изменения в классе Car:**

@Override  
**public void** move() {  
 **if** (!**rightBorder** && !**leftBorder**) {  
 **if** (**this**.**x** <= ***WIDTH*** - 440) {  
 **this**.**x** += **speed**;  
 } **else** {  
 **rightBorder** = **true**;  
 }  
 }  
 **if** (**rightBorder** && !**leftBorder**) {  
 **if** (**this**.**x** >= 0) {  
 **this**.**x** -= **speed**;  
 } **else** {  
 **leftBorder** = **true**;  
 }  
 }  
 **if** (**rightBorder** && **leftBorder**) {  
 **rightBorder** = **false**;  
 **leftBorder** = **false**;  
 }  
}

**Изменения в классе Motorbike:**

@Override  
**public void** move() {  
 **if** (!**topBorder** && !**bottomBorder**) {  
 **if** (**this**.**y** >= 0) {  
 **this**.**y** -= **speed**;  
 } **else** {  
 **topBorder** = **true**;  
 }  
 }  
 **if** (**topBorder** && !**bottomBorder**) {  
 **if** (**this**.**y** <= ***HEIGHT*** - 135) {  
 **this**.**y** += **speed**;  
 } **else** {  
 **bottomBorder** = **true**;  
 }  
 }  
 **if** (**topBorder** && **bottomBorder**) {  
 **topBorder** = **false**;  
 **bottomBorder** = **false**;  
 }  
}

**Изменения в классе MyFrame:**

**public** JButton **carAI**;  
**public** JButton **motoAI**;  
**public** CarAI **carMoving**;  
**public** MotorbikeAI **motoMoving**;  
**public static boolean** *startCarMoving*;  
**public static boolean** *startMotoMoving*;  
**public** JLabel **carMovingPriority**;  
**public** JLabel **motoMovingPriority**;  
**public** JComboBox<String> **carPriority**;  
**public** JComboBox<String> **motoPriority**;  
**public** JSlider **speedSlider**;

**speedSlider** = **new** JSlider(0,50,35);  
**speedSlider**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**speedSlider**.setBackground(**panel**.getBackground());  
JLabel speedLabel = **new** JLabel(**"Скорость: "** + Habitat.*speed*);  
**speedSlider**.addChangeListener(e -> {  
 Habitat.*speed* = (**float**) **speedSlider**.getValue() / 10;  
 speedLabel.setText(**"Скорость: "** + Habitat.*speed*);  
});  
**panel**.add(**speedSlider**);  
**panel**.add(speedLabel);  
  
**carAI** = **new** JButton(**"Движение машин"**);  
**carAI**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**carAI**.setPreferredSize(**new** Dimension(280, 30));  
**carAI**.addActionListener(e -> {  
 **synchronized** (**carMoving**.**carMonitor**) {  
 **if** (!*startCarMoving*) {  
 *startCarMoving* = **true**;  
 **carMoving**.**carMonitor**.notify();  
 } **else** {  
 *startCarMoving* = **false**;  
 }  
 }  
});  
**carAI**.setFocusable(**false**);  
**carAI**.setEnabled(**false**);  
**panel**.add(**carAI**);  
  
**motoAI** = **new** JButton(**"Движение мотоциклов"**);  
**motoAI**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**motoAI**.setPreferredSize(**new** Dimension(280, 30));  
  
**motoAI**.addActionListener(e -> {  
 **synchronized** (**motoMoving**.**motoMonitor**) {  
 **if** (!*startMotoMoving*) {  
 *startMotoMoving* = **true**;  
 **motoMoving**.**motoMonitor**.notify();  
 } **else** {  
 *startMotoMoving* = **false**;  
 }  
 }  
});  
**motoAI**.setFocusable(**false**);  
**motoAI**.setEnabled(**false**);  
**panel**.add(**motoAI**);  
  
String[] priorities = {  
 **"1"**,  
 **"2"**,  
 **"3"**,  
 **"4"**,  
 **"5"**,  
 **"6"**,  
 **"7"**,  
 **"8"**,  
 **"9"**,  
 **"10"**,  
};  
  
**carMovingPriority** = **new** JLabel(**"Приоритет машин"**);  
**carMovingPriority**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**panel**.add(**carMovingPriority**);  
  
**carPriority** = **new** JComboBox<>(priorities);  
**carPriority**.setSelectedItem(**""** + **carMoving**.getPriority());  
**panel**.add(**carPriority**);  
  
**carPriority**.addActionListener(e -> {  
 **if** (e.getSource() == **carPriority**) {  
 String prb = (String) **carPriority**.getSelectedItem();  
 **carMoving**.setPriority(Integer.*parseInt*(prb));  
 System.***out***.println(**"Приоритет машин "** + **carMoving**.getPriority());  
 }  
  
});  
**carPriority**.setFocusable(**false**);  
  
**motoMovingPriority** = **new** JLabel(**"Приоритет мотоциклов"**);  
**motoMovingPriority**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**panel**.add(**motoMovingPriority**);  
  
**motoPriority** = **new** JComboBox<>(priorities);  
**motoPriority**.setSelectedItem(**""** + **motoMoving**.getPriority());  
**panel**.add(**motoPriority**);  
  
**motoPriority**.addActionListener(e -> {  
 **if** (e.getSource() == **motoPriority**) {  
 String prb = **motoPriority**.getSelectedItem().toString();  
 **motoMoving**.setPriority(Integer.*parseInt*(prb));  
 System.***out***.println(**"Приоритет мотоциклов: "** + **motoMoving**.getPriority());  
 }  
});  
**motoPriority**.setFocusable(**false**);