МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

Отчет

по дисциплине «Технология программирования»

**Тема:** Основы программирования на Java.

Обработка событий. Механизм делегирования событий.

Вариант 8

Группа: АВТ-710

Студент: Шестак Н.П

Преподаватель: Михайленко Д.А

НОВОСИБИРСК 2020

1.1 Описание задания

1. Познакомиться с особенностями технологии Java и изучить синтаксис языка Java.
2. Изучить основные понятия и термины обработки событий в модели делегирования событий на Java (учебно-методическое пособие, главы 1 и 2).
3. Разработать учебную программу. Основная ее задача – разработка упрощенной имитации поведения объектов (все последующие лабораторные работы будут расширять это задание). Объекты реализуются через наследование: абстрактный класс + интерфейс → наследники.

Рабочий цикл программы:

* запускается процесс симуляции по клавише, генерируются объекты классов согласно заданию;
* симуляция завершается по другой клавише, выводится статистическая информация.

1. Для решения задачи:

* Разработать абстрактный класс объекта, согласно варианту индивидуального задания.
* Создать интерфейс IBehaviour, задающий поведение объекта (методы: move(), getx(), gety(), sety(), setx() и другие. Далее будут реализоваться алгоритмы движения объектов в окне программы).
* Реализовать иерархию классов, определяющих объекты по варианту и реализующие интерфейс IBehaviour.
* Создать класс Habitat (среда), определяющий размер рабочей области и хранящий массив объектов, с параметрами, заданными вариантом. Предусмотреть в классе метод Update, вызывающийся по таймеру и получающий на вход время, прошедшее от начала симуляции. В данном методе должны генерироваться новые объекты и помещаться в поле визуализации в случайном месте. Визуализация объекта - использовать готовые небольшие картинки;

Рабочее окно программы – область визуализации среды обитания объектов;

1. Симуляция должна запускаться по клавише **B** и останавливаться по клавише **E**. При остановке симуляции список очищается. Время симуляции должно отображаться текстом в области визуализации и скрываться/показываться по клавише **T**;
2. По завершению симуляции область отображения объектов очищается и в поле визуализации выводится информация о количестве и типе сгенерированных объектов, а также время симуляции. Текст должен быть форматирован, т.е. выводиться с использованием разных шрифтов и цветов.
3. Параметры симуляции задаются в классе Habitat.

1.2 Исходные данные, соответствующие варианту

*Вариант 8*

|  |
| --- |
| Список транспортных средств на дороге состоит из двух категорий: автомобили и мотоциклы. Автомобили генерируются каждые N1 секунд с вероятностью P1. Мотоциклы генерируются каждые N2 секунд с вероятностью P2. |

2. Ход работы

Для реализации программы было создано несколько классов:

|  |  |
| --- | --- |
| Class | Properties |
| Automobile, Motocycle, Terrain | Наследники класса JImage, хранящие свойства изображения необходимые для корректного вывода его на экран программы |
| Habitat | Один из основных классов, определяющий размер рабочей области и хранящий массив объектов, с параметрами, заданными вариантом |
| IBehaviour | Класс для задания интефейса |
| JImage | Класс, используемый для вывода объектов на экран, использующий элементы интерфейса IBehaviour |
| Main | Класс, создающий среду и используемый для работы с программой при помощи клавиш |

Для каждого из классов было реализовано несколько основных методов:  
Классу **Main** был необходим функционал для взаимодействий при помощи клавиатуры, было решено использовать keyListener:

|  |
| --- |
| public void keyPressed(KeyEvent e) {  System.*out*.println("TIK-TAK");  switch (e.getKeyChar()) {  case 'b':  System.*out*.println("b");  theUniverse.timerForAutomobile.start();  theUniverse.timerForMotocycle.start();  break;  ... |

У класса **Habitat** методы update – для получения времени симуляции и habitat – для определения размера рабочей области.

|  |
| --- |
| public void update(long timeFromStart, boolean is\_automobile) {  JTimer.setText(String.*valueOf*(new Date().getTime() - timeFromStart));  System.*out*.println("update");  if (is\_automobile) {  int tempX = (int) (VehicleWidth \* Math.*random*());  int tempY = (int) (VehicleHeigth \* Math.*random*());  addVehicle(new Automobile(tempX, tempY));  } else  ... |

|  |
| --- |
| public Habitat() {  setSize(FrameWidth, FrameHeigth);  setContentPane(new JPanel());  add(new Terrain(0, 0));  setLayout(null);  setVisible(true);  JTimer.setBounds(10, 10, 300, 100);  setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  } |

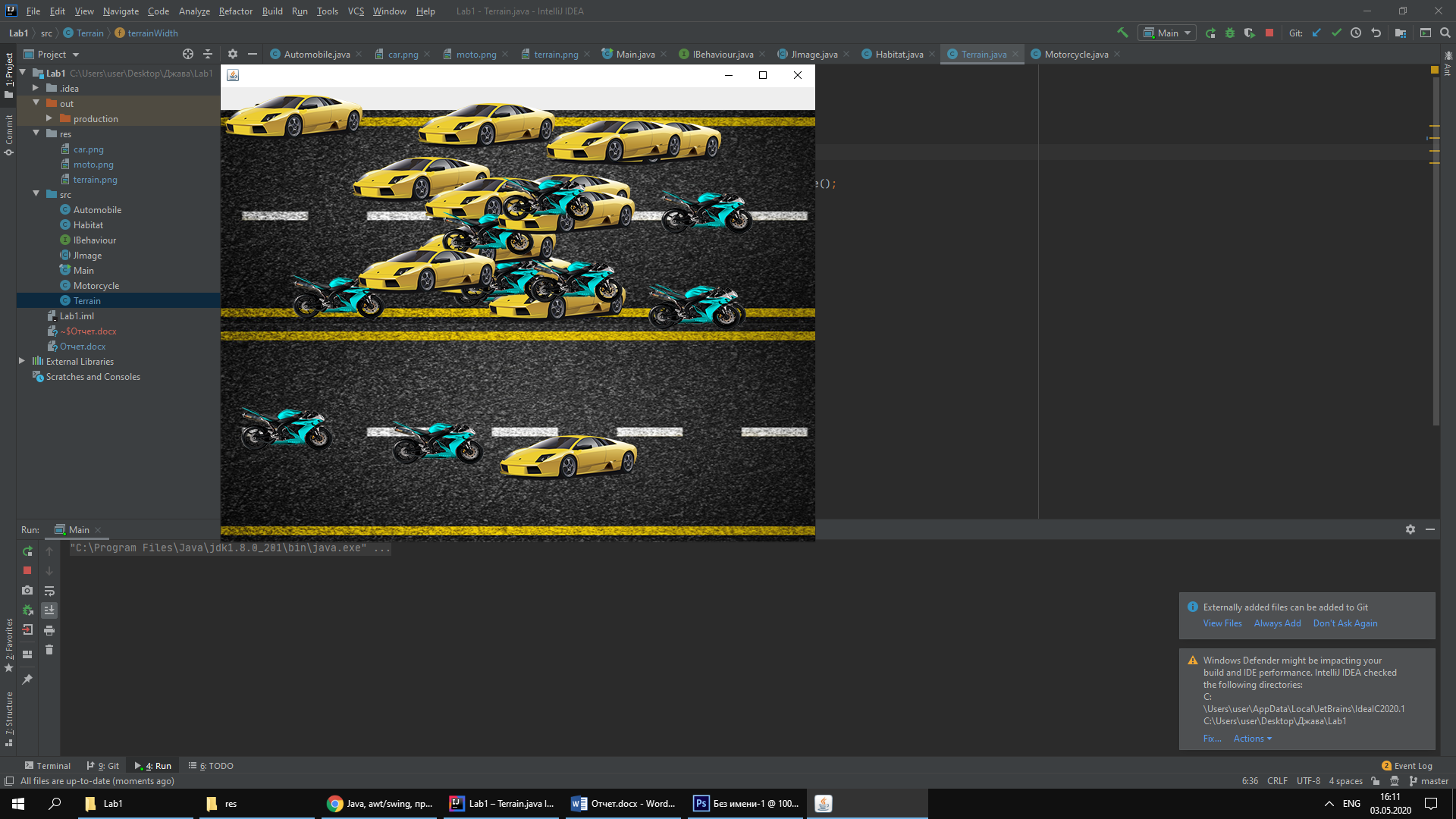
Класс **JImage** использует инструменты интерфейса при помощи одноименного метода

|  |
| --- |
| public JImage(){  setSize(getWidth(), getHeight());  setBounds(getX(), getY(), getWidth(), getHeight());  setLayout(null);  } |

3. Результаты выполнения работы

Готовая программа выполняет все необходимые функции, такие как:

* Генерация объектов и вывод их в рабочую область



*Рисунок 1 Генерация объектов*

* Возможность скрыть/показать таймер

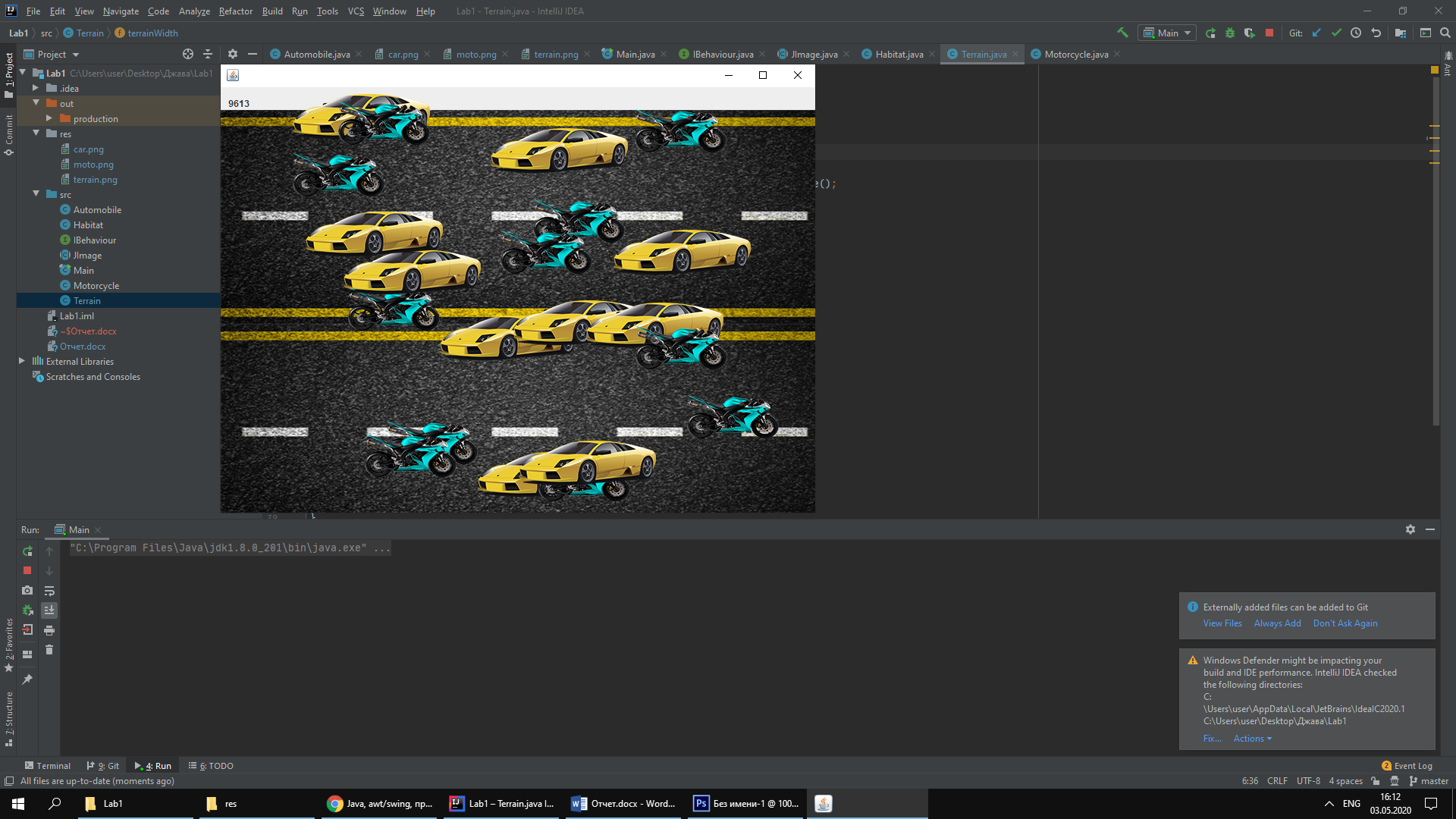
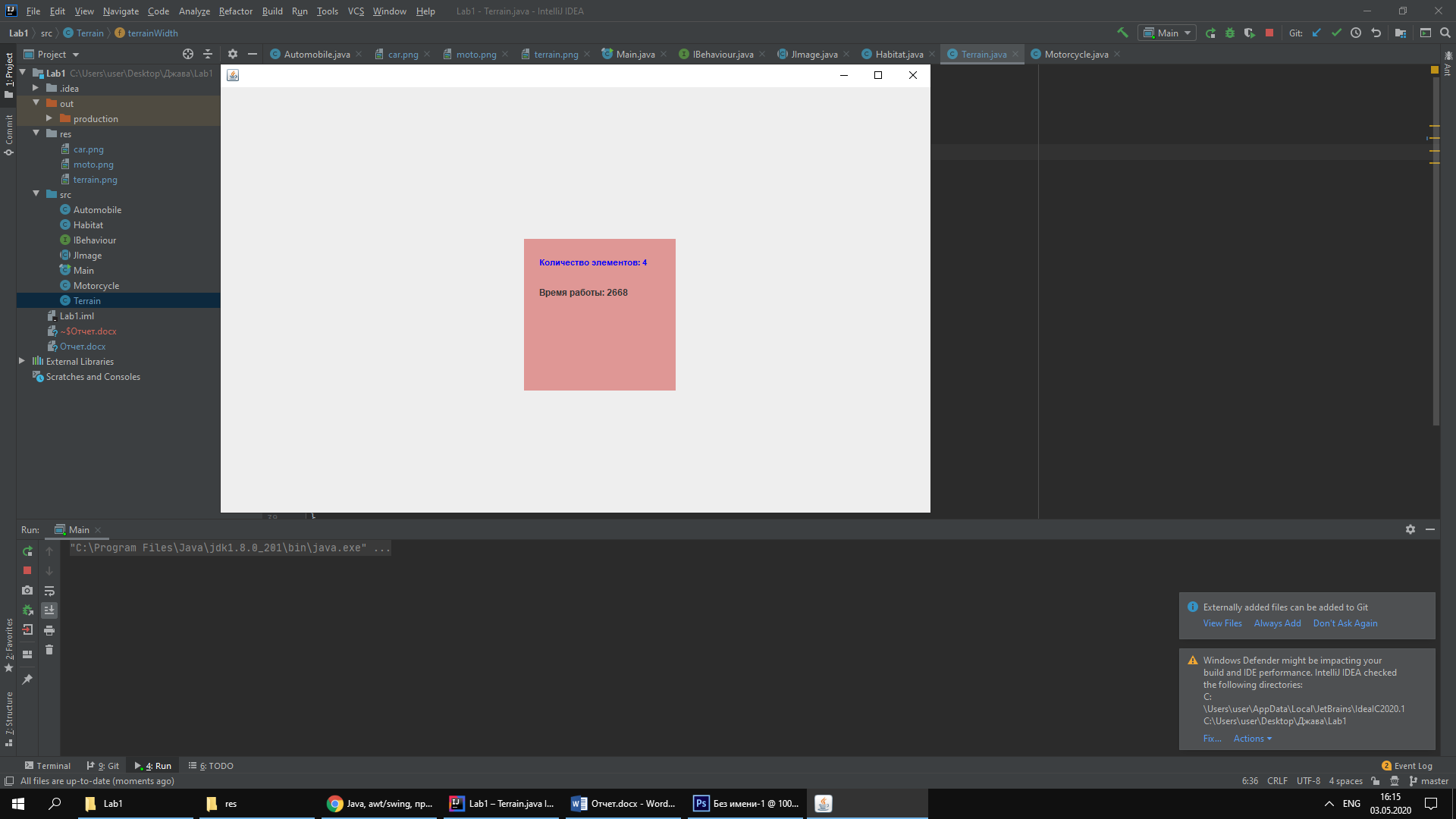


Рисунок 2 Наличие таймера

* Вывод итоговой информации по окончанию симуляции



*Рисунок 3 Итоговая информация*

1. Выводы

В ходе выполнения лабораторной работы была реализована программа в соответствии с вариантом задания. Были изучены основные основные понятия и термины обработки событий в модели делегирования событий на Java.