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 №3

По дисциплине: «Технологии и методы программирования»

Классы-коллекции

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 классов-коллекций в Java.

**Задание к лабораторной работе:**

Доработать программу, созданную в лабораторной работе № 2:

1. добавить генерируемым объектам понятия «время рождения» и «время жизни». Время рождения устанавливается в момент генерации объекта, и по значению соответствует времени, прошедшему от начала симуляции. Время жизни – время, через которое объект должен исчезнуть, считая от времени рождения;
2. вынести установку параметров времени жизни объектов в пользовательский интерфейс. Для каждого типа объекта должно задаваться собственное время. Рекомендуется использовать текстовые поля, но следуют помнить о проверке на ввод некорректных данных;
3. добавить генерируемым объектам уникальные целочисленные идентификаторы (случайные числа), которые назначаются при генерации объекта. Для хранения сгенерированных идентификаторов используйте коллекцию удобную для поиска по варианту;
4. использовать коллекции по варианту. При генерации объекта происходит добавление его в коллекцию (в класс добавить поле идентификатора), а во вторую коллекцию: идентификаторы существующих объектов, в третью идентификатор ̶ время рождения. При возникновении события по таймеру обойдите коллекцию и удалите все объекты, время жизни которых истекло, а также все данные во вспомогательных коллекциях.
5. добавьте в панель управления кнопку «Текущие объекты». По нажатию на эту кнопку появляется модальное диалоговое окно, содержащее список всех «живых» объектов на момент нажатия со временем их рождения (время рождения – ключ). В класс диалогового окна должна передаваться коллекция с хранением объектов по времени рождения. Типы коллекций задаются вариантом.

*Вариант 8*

Коллекция для хранения объектов: ArrayList

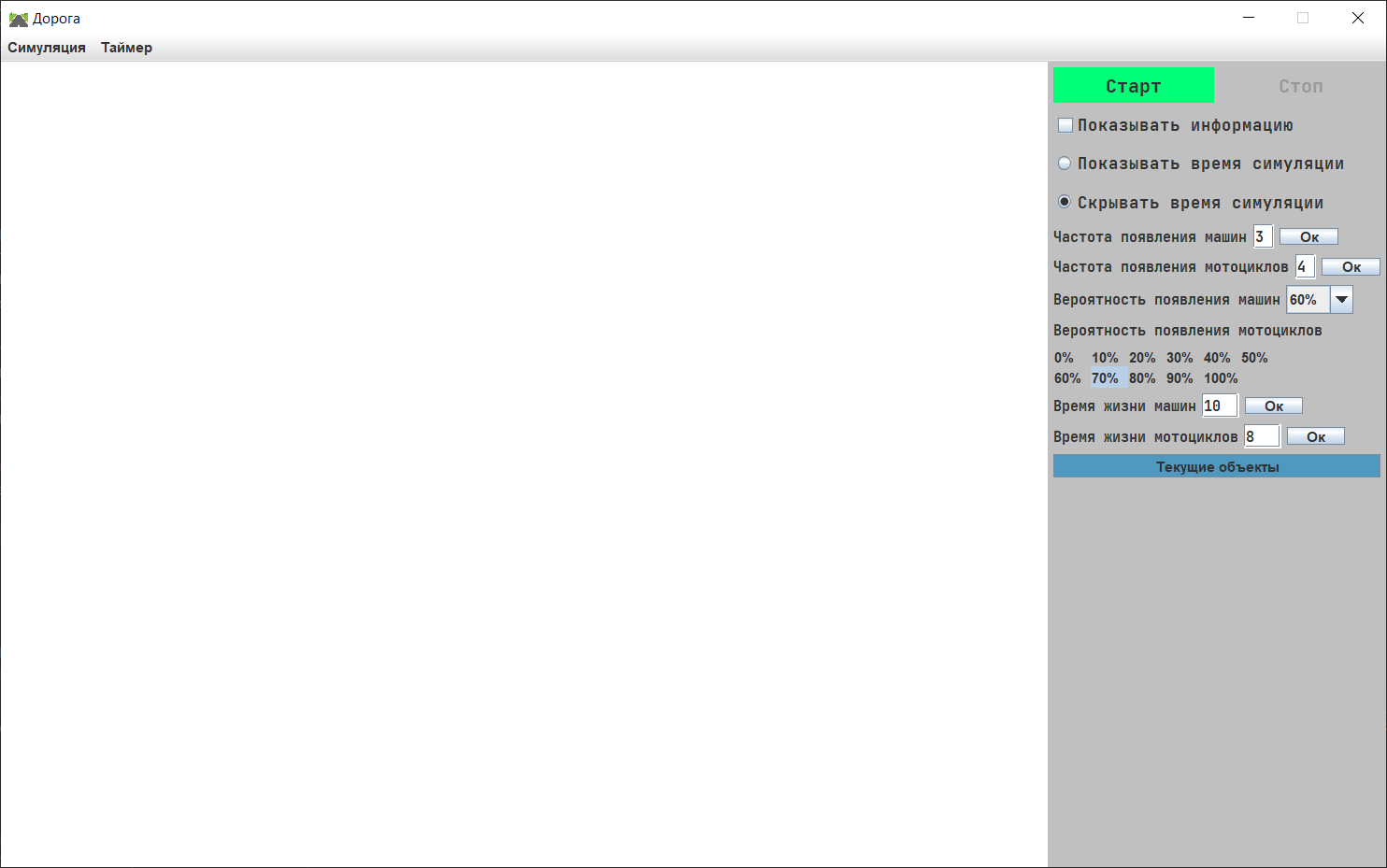
Коллекция для хранения и поиска уникальных идентификаторов: HashSet

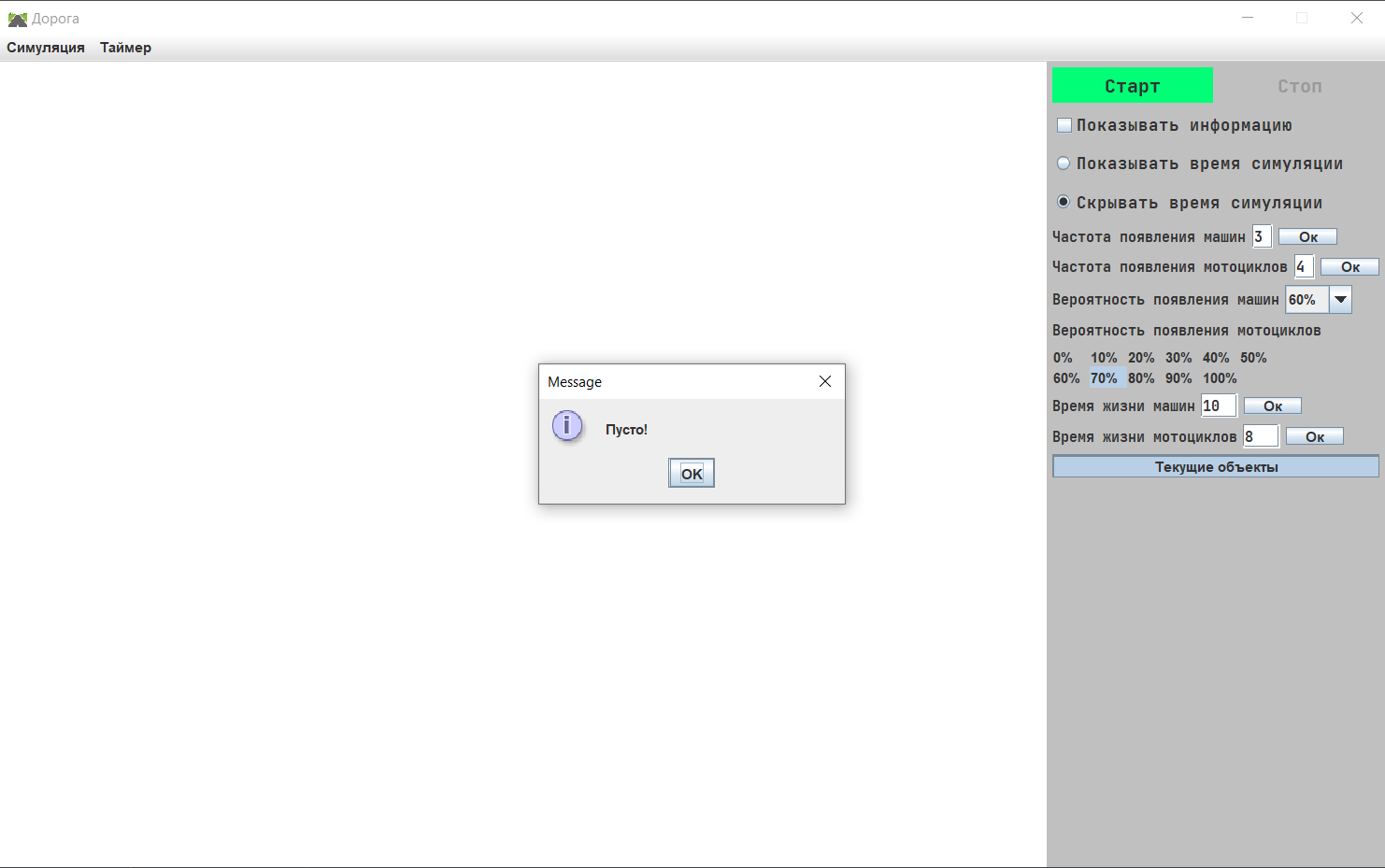
Коллекция для хранения времени рождения объектов: TreeMap

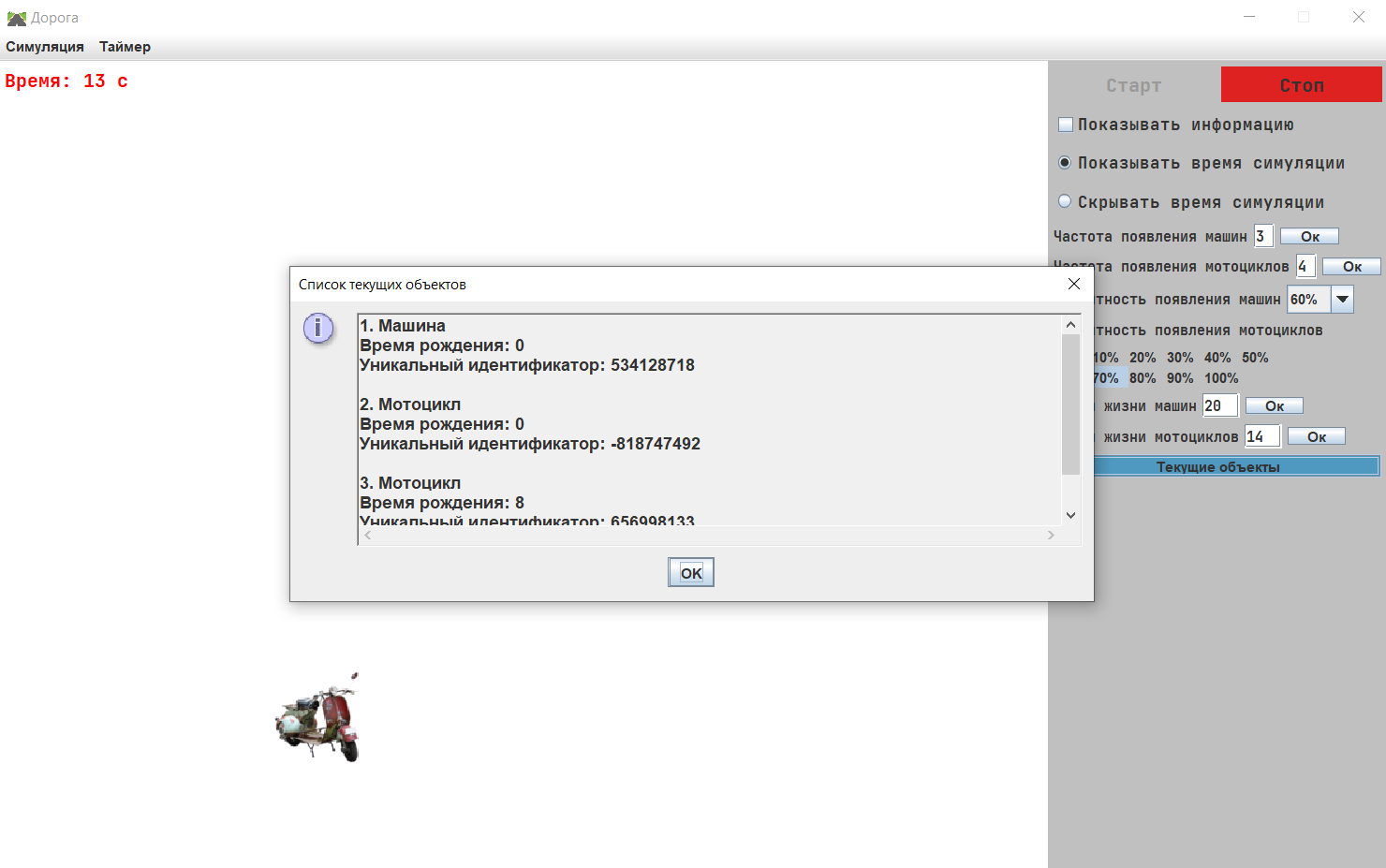
**Выполнение работы:**

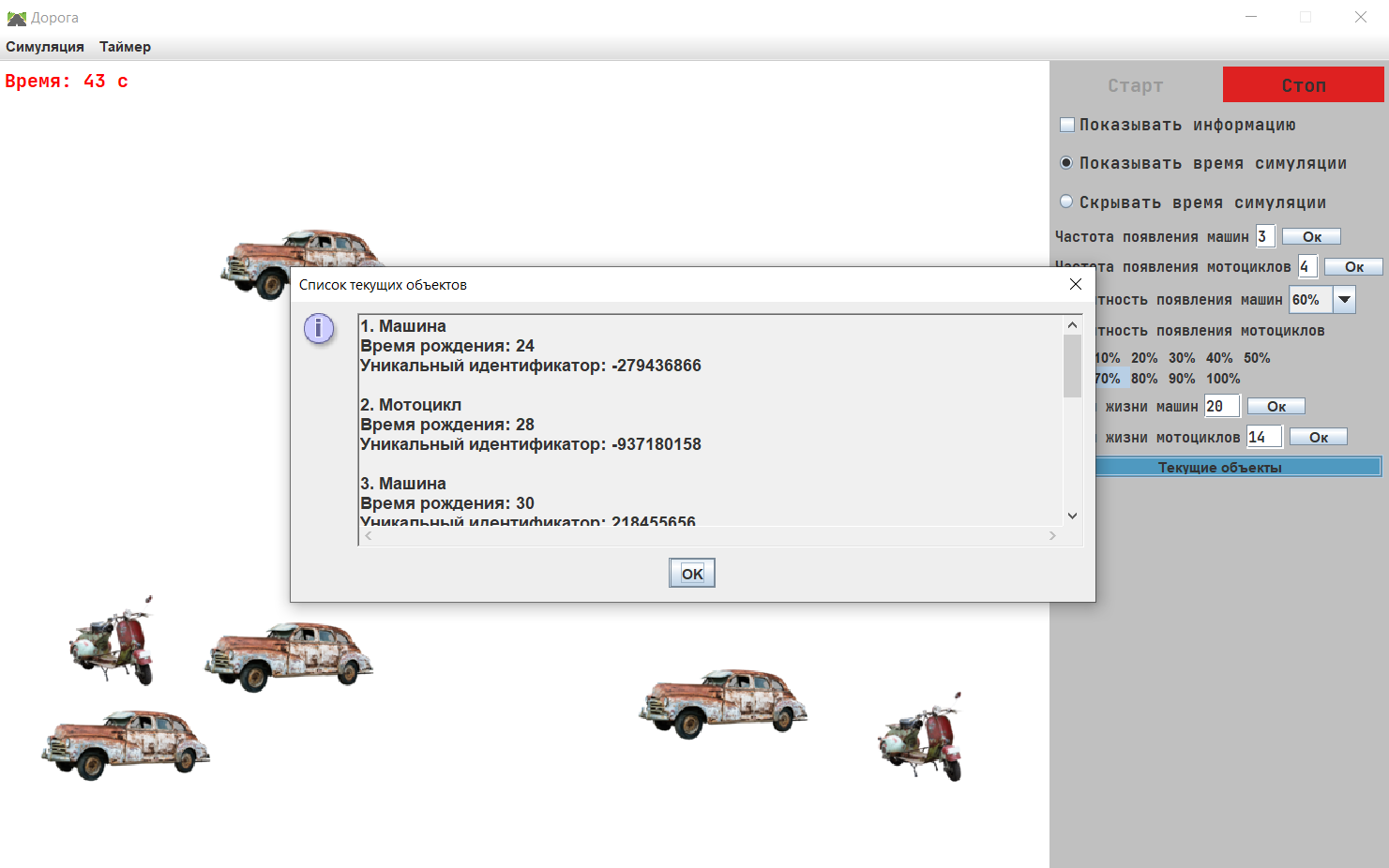
В абстрактный класс Vehicle были добавлены поля id и timeAppear, а также их геттеры и сеттеры. Классы Car и Motorbike теперь имеют собственные статические поля timeLifeCar и timeLifeMoto соответственно, которые определяют время жизни объектов. Они используются в классе Habitat для удаления объектов, там же реализованы интерфейсы List, Set, Map классами ArrayList, HashSet и TreeMap соответственно. В классе MyFrame был дополнен пользовательский интерфейс. Теперь пользователь может управлять временем жизни объектов 2 типов и получать актуальную информацию о сгенерированных объектах с помощью соответствующей кнопки.

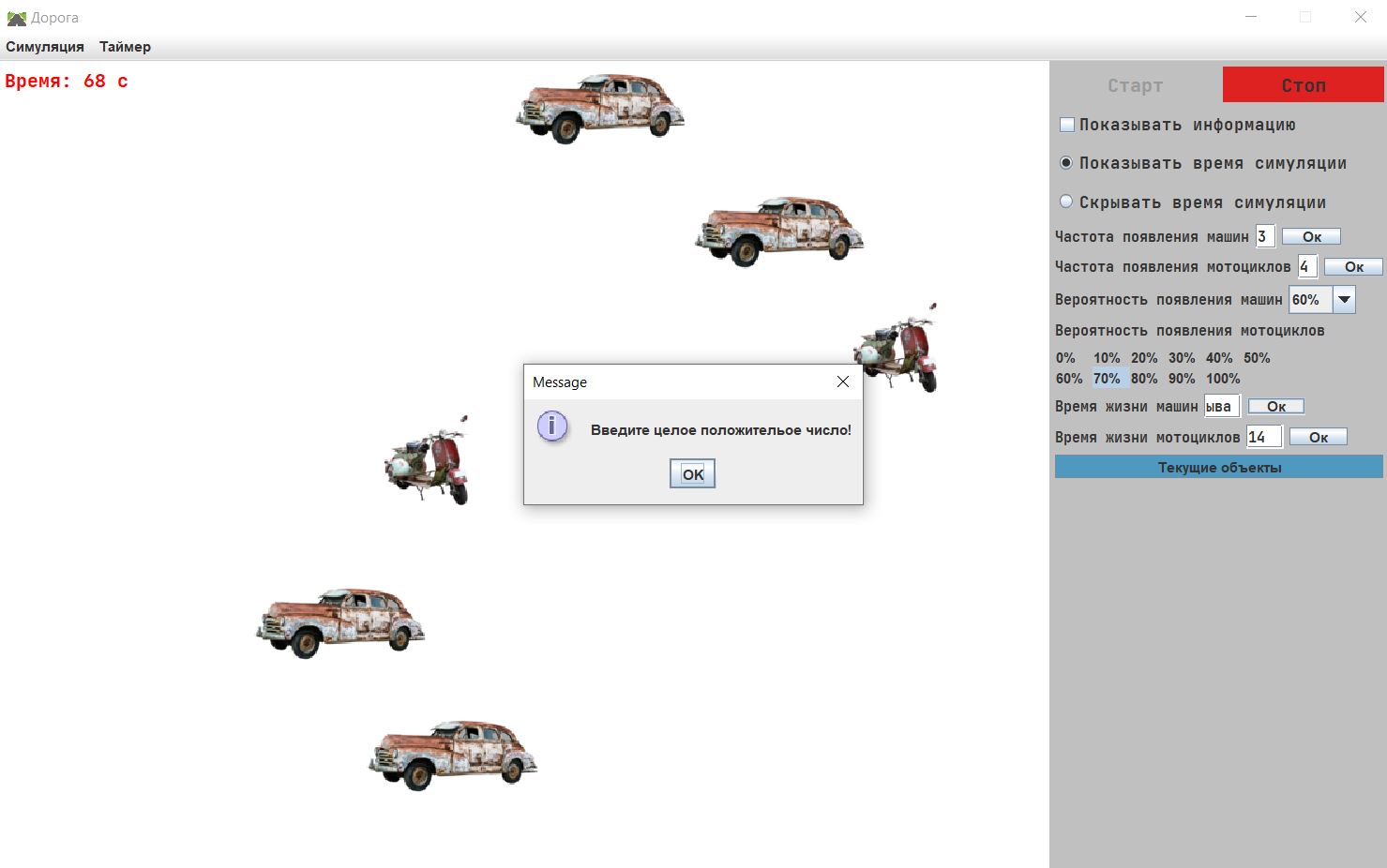
**Пример выполнения работы**



****

****





**Вывод:** в ходе выполнения лабораторной работы были изучены особенности реализации классов-коллекций в Java. В программу были добавлены необходимые реализации интерфейсов коллекций. Также пользовательский интерфейс пополнился новыми функциями.

**Изменения в классе MyFrame:**

**public** JButton **submitCarTime**;  
**public** JButton **submitMotoTime**;  
**public** JTextField **carsTimeText**;  
**public** JTextField **motoTimeText**;  
**public** JButton **currentVehicles**;

**submitCarTime** = **new** JButton(**"Ок"**);  
**submitCarTime**.setPreferredSize(**new** Dimension(50, 15));  
JLabel carsTime = **new** JLabel(**"Время жизни машин"**);  
**carsTimeText** = **new** JTextField(**""** + Car.*getTimeLifeCar*(), 4);  
carsTime.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**carsTimeText**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**submitCarTime**.addActionListener(e -> {  
 **try** {  
 Car.*setTimeLifeCar*(Integer.*parseInt*(**carsTimeText**.getText()));  
 **if** (Car.*getTimeLifeCar*() <= 0) **throw new** Exception();  
 } **catch** (Exception exp) {  
 JOptionPane.*showMessageDialog*(**this**, **"Введите целое положительое число!"**);  
 Car.*setTimeLifeCar*(10);  
 System.***out***.println(**"Поймано исключение "** + exp.getMessage());  
 **carsTimeText**.setText(**""** + Car.*getTimeLifeCar*());  
 }  
});  
**submitCarTime**.setFocusable(**false**);  
**panel**.add(carsTime);  
**panel**.add(**carsTimeText**);  
**panel**.add(**submitCarTime**);  
  
**submitMotoTime** = **new** JButton(**"Ок"**);  
**submitMotoTime**.setPreferredSize(**new** Dimension(50, 15));  
JLabel motoTime = **new** JLabel(**"Время жизни мотоциклов"**);  
**motoTimeText** = **new** JTextField(**""** + Motorbike.*getTimeLifeMoto*(), 4);  
motoTime.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**motoTimeText**.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 12));  
**submitMotoTime**.addActionListener(e -> {  
 **try** {  
 Motorbike.*setTimeLifeMoto*(Integer.*parseInt*(**motoTimeText**.getText()));  
 **if** (Motorbike.*getTimeLifeMoto*() <= 0) **throw new** Exception();  
 } **catch** (Exception exp) {  
 JOptionPane.*showMessageDialog*(**this**, **"Введите целое положительое число!"**);  
 Motorbike.*setTimeLifeMoto*(8);  
 System.***out***.println(**"Поймано исключение "** + exp.getMessage());  
 **motoTimeText**.setText(**""** + Motorbike.*getTimeLifeMoto*());  
 }  
});  
**submitMotoTime**.setFocusable(**false**);  
**panel**.add(motoTime);  
**panel**.add(**motoTimeText**);  
**panel**.add(**submitMotoTime**);  
  
**currentVehicles** = **new** JButton(**"Текущие объекты"**);  
**currentVehicles**.setPreferredSize(**new** Dimension(280, 20));  
**currentVehicles**.setBackground(**new** Color(79, 153, 192));  
**currentVehicles**.addActionListener(e -> {  
 StringBuilder currentObjects = **new** StringBuilder();  
 **for** (**int** i = 0; i < Habitat.***vehicles***.size(); i++) {  
 Vehicle v = Habitat.***vehicles***.get(i);  
 **if** (v **instanceof** Car) {  
 currentObjects.append(i + 1).  
 append(**". Машина\nВремя рождения: "**).append(Habitat.***times***.get(v.getId())).  
 append(**"\nУникальный идентификатор: "**).append(v.getId()).append(**"\n\n"**);  
 } **else** {  
 currentObjects.append(i + 1).  
 append(**". Мотоцикл\nВремя рождения: "**).append(Habitat.***times***.get(v.getId())).  
 append(**"\nУникальный идентификатор: "**).append(v.getId()).append(**"\n\n"**);  
 }  
 }  
 **if** (currentObjects.length() == 0) {  
 JOptionPane.*showMessageDialog*(**this**, **"Пусто!"**);  
 } **else** {  
 TextArea stats = **new** TextArea(currentObjects.toString());  
 stats.setEditable(**false**);  
 stats.setFont(**new** Font(**"JetBrains Mono"**, Font.***BOLD***, 14));  
 JOptionPane.*showMessageDialog*(**this**, stats, **"Список текущих объектов"**, JOptionPane.***INFORMATION\_MESSAGE***);  
 }  
});  
**currentVehicles**.setFocusable(**false**);  
**panel**.add(**currentVehicles**);

**Изменения в классе Habitat:**

**public final static** List<Vehicle> ***vehicles*** = **new** ArrayList<>();  
**public static** Set<Integer> *ids* = **new** HashSet<>();  
**public final static** Map<Integer, Long> ***times*** = **new** TreeMap<>();

**public void** update(**long** time) {  
 **for** (**int** i = 0; i < ***vehicles***.size(); i++) {  
 Vehicle v = ***vehicles***.get(i);  
 **if** ((v **instanceof** Car) &&  
 (time - v.getTimeAppear() >= Car.*getTimeLifeCar*())) {  
 ***vehicles***.remove(v);  
 *ids*.remove(v.getId());  
 ***times***.remove(v.getId());  
 i--;  
 System.***out***.println(**"-car{"** + v.getX() + **"; "** + v.getY() + **"; "** + v.getId() + **"}"**);  
 } **else if** ((v **instanceof** Motorbike) &&  
 (time - v.getTimeAppear() >= Motorbike.*getTimeLifeMoto*())) {  
 ***vehicles***.remove(i);  
 *ids*.remove(i);  
 ***times***.remove(i);  
 i--;  
 System.***out***.println(**"-motorbike{"** + v.getX() + **"; "** + v.getY() + **"; "** + v.getId() + **"}"**);  
 }  
 }  
  
 **if** (**random**.nextInt(100) < **P1** && (**int**) time % **N1** == 0) {  
 Car car = **new** Car(**random**.nextInt(**WIDTH**-440), **random**.nextInt(**HEIGHT**-128));  
 car.setTimeAppear(time);  
 ***vehicles***.add(car);  
 ***times***.put(car.getId(), time);  
 System.***out***.println(**"+car{"** + car.getX() + **"; "** + car.getY() + **"; "** + car.getId() + **"}"**);  
 **carCount**++;  
 }  
 **if** (**random**.nextInt(100) < **P2** && (**int**) time % **N2** == 0) {  
 Motorbike motorbike = **new** Motorbike(**random**.nextInt(**WIDTH**-390), **random**.nextInt(**HEIGHT**-144));  
 motorbike.setTimeAppear(time);  
 ***vehicles***.add(motorbike);  
 ***times***.put(motorbike.getId(), time);  
 System.***out***.println(**"+motorbike{"** + motorbike.getX() + **"; "** + motorbike.getY() + **"; "** + motorbike.getId() + **"}"**);  
 **motoCount**++;  
 }  
  
}

**Изменения в классе Vehicle:**

**public int x**, **y**, **id**;  
**public long timeAppear**;

**public long** getTimeAppear() {**return timeAppear**;}  
  
**public void** setTimeAppear(**long** timeAppear) {**this**.**timeAppear** = timeAppear;}  
  
**public int** getId() {**return id**;}

**Изменения в классе Car:**

**public class** Car **extends** Vehicle **implements** IBehaviour {  
 **private static long** *timeLifeCar* = 10;  
  
 **public** Car(**int** x, **int** y) {  
 **this**.**image** = **new** ImageIcon(**"JavaLabs/src/nstu/imgs/car.png"**);  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **int** id = **new** Random().nextInt(2000000000) - 1000000000;  
 **while** (**true**) {  
 **if** (Habitat.*ids*.contains(id)) {  
 id = **new** Random().nextInt(2000000000) - 1000000000;  
 } **else** {  
 **break**;  
 }  
 }  
 **this**.**id** = id;  
 Habitat.*ids*.add(id);  
 }  
  
 **public static long** getTimeLifeCar() {  
 **return** *timeLifeCar*;  
 }  
  
 **public static void** setTimeLifeCar(**long** timeLifeCar) {  
 Car.*timeLifeCar* = timeLifeCar;  
 }  
}

**Изменения в классе Motorbike:**

**public class** Motorbike **extends** Vehicle **implements** IBehaviour {  
 **private static long** *timeLifeMoto* = 8;  
  
 **public** Motorbike(**int** x, **int** y) {  
 **this**.**image** = **new** ImageIcon(**"JavaLabs/src/nstu/imgs/moto.png"**);  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **int** id = **new** Random().nextInt(2000000000) - 1000000000;  
 **while** (**true**) {  
 **if** (Habitat.*ids*.contains(id)) {  
 id = **new** Random().nextInt(2000000000) - 1000000000;  
 } **else** {  
 **break**;  
 }  
 }  
 **this**.**id** = id;  
 Habitat.*ids*.add(id);  
 }  
  
 **public static long** getTimeLifeMoto() {  
 **return** *timeLifeMoto*;  
 }  
  
 **public static void** setTimeLifeMoto(**long** timeLifeMoto) {  
 Motorbike.*timeLifeMoto* = timeLifeMoto;  
 }  
}