

Лабораторная работа №3

Технология программирования

*«Классы-коллекции»*

Студент: Воликов М.В.

Группа: АВТ-910

Вариант: 4

Преподаватель: Дубков И.С.



НОВОСИБИРСК 2021

Оглавление

[Задание 3](#_Toc38167595)

[Ход работы 3](#_Toc38167596)

[Вывод 3](#_Toc38167597)

# Задание

1. Изучить особенности реализации классов-коллекций в Java.
2. Доработать программу, созданную в лабораторной работе № 2:
3. 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;
4. 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;
5. добавить генерируемым объектам уникальные целочисленные идентификаторы (случайные числа), которые назначаются при генерации объекта. Для хранения сгенерированных идентификаторов используйте коллекцию удобную для поиска по варианту;
6. 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; Для обеспечения доступа к коллекциям используйте структурный паттерн Singleton.
7. добавьте в панель управления кпопку «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 4***

Коллекция для хранения объектов: Vector

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

# Ход работы

## Структурный паттерн Singleton

В первую очередь был изменён класс Singleton. Он был расширен путём добавления в него классов-коллекций.

public class Singleton {  
 private Singleton(){}  
 private static Vector<Rabbit> *vector* = new Vector<>();  
 public static Vector<Rabbit> getVector() { return *vector*; }  
 private static TreeSet<Long> *treeSetID* = new TreeSet<>();  
 public static TreeSet<Long> getTreeSetID() {return *treeSetID*; }  
 private static HashMap<Long, Long> *hashMap* = new HashMap<>();  
 public static HashMap<Long, Long> getHashMap() { return *hashMap*; }  
}

## Классы Rabbit, Albino и Ordinary

В абстрактный класс Rabbit были добавлены поля ID и BirthTime, так как эти поля одинаковы для его наследников. В свою очередь в классы Albino и Ordinary было добавлено статическое поле TimeOfLife (время жизни), так как для альбиносов и обыкновенных кроликов эти значения отличаются.

## Новые объекты интерфейса

Интерфейс был дополнен поясняющими метками. Для задания времени жизни были созданы текстовые поля. Размещение и добавление слушателей реализовано аналогично предыдущей лабораторной работе.

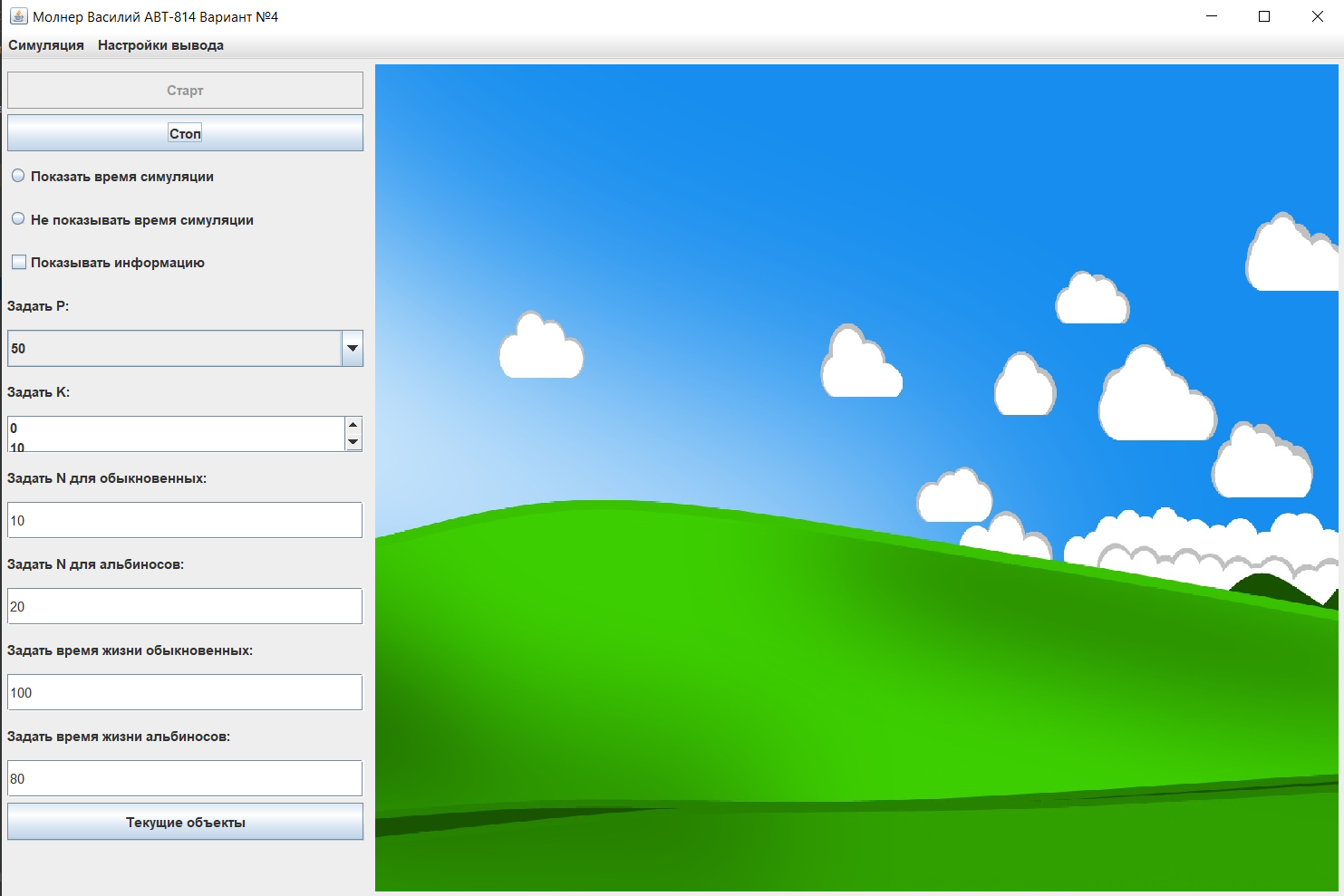


Рисунок - Главное окно

## Модальное окно объектов

Окно объектов вызывается по соответствующей кнопке. После этого запускается метод ActionShowObjects():

private void ActionShowObjects() {  
 String string = new String();  
 for (Rabbit rabbit : Singleton.*getVector*()) {  
 string += (rabbit.ID + " " + rabbit.BirthTime + "\n");  
 }  
 jTextAreaObjects.setText("Ключ/Время рождения\n" + string);  
 jDialogObjects.setVisible(true);  
}

Предварительно окно создаётся в конструкторе класса GUI.

jDialogObjects.setSize(300, 400);  
jDialogObjects.setLocation(350,200);  
jTextAreaObjects.setEditable(false);  
jDialogObjects.add(jTextAreaObjects);

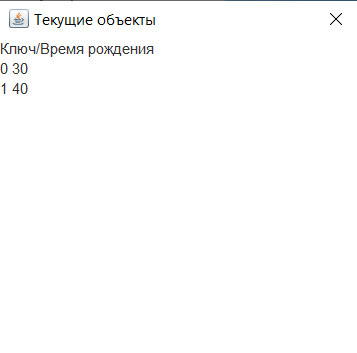


Рисунок - Модальное окно "Текущие объекты"

## Класс Habitat

В классе Habitat в метод Update был добавлен обход текущих объектов и последующее их удаление исходя из условия «Текущее время > Время жизни + Время рождения».

for (int i = 0; i < Singleton.*getVector*().size(); i++) {  
 if (Singleton.*getVector*().get(i) instanceof Ordinary) {  
 if (time == Singleton.*getVector*().get(i).BirthTime + Ordinary.*TimeOfLife*) {  
 Rabbit.*AllQuantity*--;  
 Ordinary.*OrdinaryQuantity*--;  
 Singleton.*getVector*().remove(i);  
 }  
 } else {  
 if (time == Singleton.*getVector*().get(i).BirthTime + Albino.*TimeOfLife*) {  
 Rabbit.*AllQuantity*--;  
 Albino.*AlbinoQuantity*--;  
 Singleton.*getVector*().remove(i);  
 }  
 }  
}

# Вывод

В ходе лабораторной работы были изучены различные классы-коллекции, а также они были применены на практике. Был расширен интерфейс программы.