МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НГТУ

Лабораторная работа №3

по предмету «Технология программирования»

на тему «Классы-коллекции»

Факультет: АВТ Преподаватель:

Группа: АВТ-809 Михайленко Д.А.

Студент: Швыдко Е.А.

Новосибирск 2020

Цели работы:

***Вариант 11***

Коллекция для хранения объектов: ArrayList

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения времени рождения объектов: HashMap

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.

Ход разработки:

* Были доработаны классы Cat и Dog: добавлены поля «время рождения», «время жизни», «ID»

**public** **class** Cat **extends** Animal {

**private** **static** BufferedImage Image;

**static** {

**try** {

Image = ImageIO.read(**new** **File**("src/com/company/assets/Cat.png"));

} **catch** (**IOException** e) {

e.printStackTrace();

}

}

**public** Cat(int \_X, int \_Y, int \_ID){

**this**.X = \_X;

**this**.Y = \_Y;

**this**.id = \_ID;

}

**public** int getID() {**return** id;}

**@Override**

**public** int getX() {

**return** **super**.getX();

}

**@Override**

**public** int getY() {

**return** **super**.getY();

}

**@Override**

**public** void setX(int X) {

**this**.X = X;

}

**@Override**

**public** void setY(int Y) {

**this**.Y = Y;

}

**@Override**

**public** void move(){

}

**public** void draw(Graphics g){

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(Image, **this**.X, **this**.Y, 80, 80, **null**);

}

}

* Были доработаны текстовые поля, позволяющие редактировать параметры симуляции, а именно были добавлены проверки на правильность вводимых данных

dogLifeLabel = new JLabel("�'ремя жизни соб.");

add(dogLifeLabel);

dogLifeLabel.setBounds(790,10,120,13);

catLifeLabel = new JLabel("�'ремя жизни кош.");

add(catLifeLabel);

catLifeLabel.setBounds(915,10,120,13);

dogLifeField = new JTextField(10);

add(dogLifeField);

dogLifeField.setBounds(790,25,120,20);

dogLifeField.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent actionEvent) {

double kk = **Double**.parseDouble(dogLifeField.getText());

**if**(kk > 0) Habitat.dogTTL = kk;

**else** {

Habitat.dogTTL = 1.0;

dogLifeField.setText("1.0");

}

}

});

catLifeField = new JTextField(10);

add(catLifeField);

catLifeField.setBounds(915,25,120,20);

catLifeField.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent actionEvent) {

double kk = **Double**.parseDouble(catLifeField.getText());

**if**(kk > 0) Habitat.catTTL = kk;

**else** {

Habitat.catTTL = 1.0;

catLifeField.setText("1.0");

}

}

});

* Был доработан класс Habitat, для проверки «валидности» объектов. Для доступа к коллекциям был использован паттерн Singleton

**public** void update(double time){

**if**(**Math**.random() < catProb) {

**if**(time % catDelay == 0){

int \_id = (int) (**Math**.random() \* 10000000);

factory = createObjects(**true**);

Singleton.getVector().getArray().add(factory.createObject(\_id));

Singleton.getVector().getIds().add(\_id);

Singleton.getVector().getTTL().put(\_id, time);

**System**.out.println("Кошка");

cats++;

repaint();

}

}

**else** {

**if**(time % dogDelay == 0) {

int \_id = (int) (**Math**.random() \* 10000000);

factory = createObjects(**false**);

Singleton.getVector().getArray().add(factory.createObject(\_id));

Singleton.getVector().getIds().add(\_id);

Singleton.getVector().getTTL().put(\_id, time);

**System**.out.println("Собака");

dogs++;

repaint();

}

}

**for**(int i = 0; i < Singleton.getVector().getArray().size(); i++){

**if** (Singleton.getVector().getIds().contains(Singleton.getVector().getArray().get(i).getID())){

double timeOfBorn = Singleton.getVector().getTTL().get(Singleton.getVector().getArray().get(i).getID());

double ttl;

**if**(Singleton.getVector().getArray().get(i) **instanceof** Cat) ttl = catTTL;

**else** ttl = dogTTL;

**if** ((time - timeOfBorn) >= ttl) {

**if** (Singleton.getVector().getArray().get(i) **instanceof** Dog) {

dogs--;

**System**.out.println("-Собака");

}

**else** {

cats--;

**System**.out.println("-Кошка");

}

Singleton.getVector().getIds().remove(Singleton.getVector().getArray().get(i).getID());

Singleton.getVector().getTTL().remove(Singleton.getVector().getArray().get(i).getID());

Singleton.getVector().getArray().remove(i);

repaint();

}

}

}

}

* В панель элементов управления была добавлена кнопка «Objects now», по нажатию на которую появляется модальное диалоговое окно с таблицей «живых» объектов. Ниже описан класс, реализующий этот функционал:

**public** **class** ObjectsDialog **extends** JDialog {

ObjectsDialog(MFrame parent) {

**super**(parent, "Текущие объекты", **true**);

setSize(300,300);

setLayout(**new** FlowLayout());

setResizable(**false**);

setLocationRelativeTo(parent);

JPanel panel = **new** JPanel();

panel.setLayout(**new** FlowLayout());

JTextArea objs = **new** JTextArea(25,50);

JScrollPane scrollPane = **new** JScrollPane(objs);

scrollPane.setPreferredSize(**new** Dimension(270,250));

**String** msg ="Класс\t��"\t�'ремя рождения"***;***

**for** (Animal s : Singleton.getVector().getArray()) {

**if** (s **instanceof** Dog) {

msg += "\nСобака\t" + ((Dog)s).getID() + "\t" + Singleton.getVector().getTTL().get(((Dog)s).getID());

} **else** **if** (s **instanceof** Cat) {

msg += "\nКошка\t" + ((Cat)s).getID() + "\t" + Singleton.getVector().getTTL().get(((Cat)s).getID());

}

}

Font font = objs.getFont();

objs.setFont(font.deriveFont(font.getSize()+3.5f));

objs.setText(msg);

objs.setEditable(**false**);

panel.add(scrollPane);

setContentPane(panel);

setVisible(**true**);

}

}

Выводы:

В результате выполнения работы были рассмотрены классы-коллекции, упрощающие управление большим количеством объектов.