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 «Технология программирования»

Тема: «Основы программирования на Java.

Обработка событий. Механизм делегирования событий. Разработка графического интерфейса приложения.»

Вариант №2

Выполнил:

студент группы АВТ-716

Тесленко Н.Б.

Проверил:

Михайленко Дмитрий Анатольевич

Новосибирск, 2020 г.

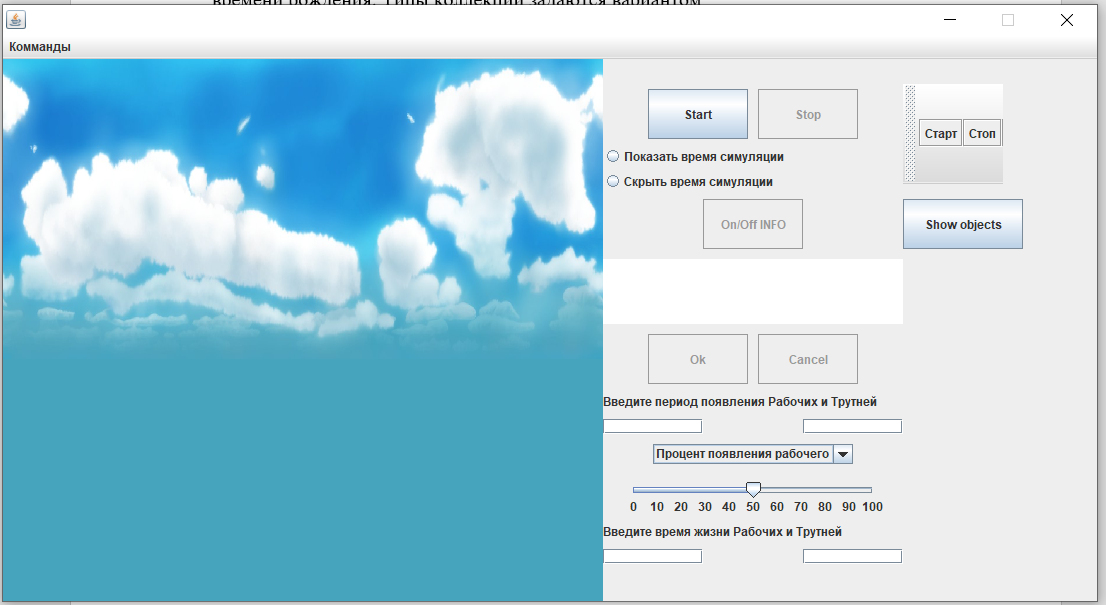
#### Практические задания

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;

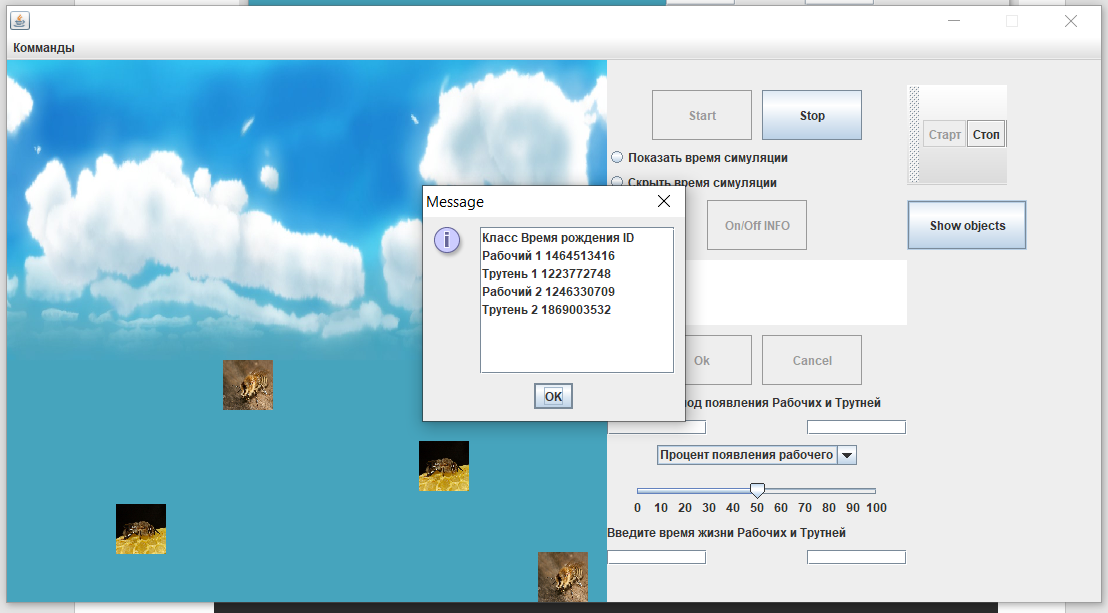
добавьте в панель управления кпопку «Текущие объекты». По нажатию на эту кнопку появляется модальное диалоговое окно, содержащее список всех «живых» объектов на момент нажатия со временем их рождения (тип объекта, время рождения, идентификатор). В класс диалогового окна должна передаваться коллекция с хранением объектов по времени рождения. Типы коллекций задаются вариантом

**Результаты работы программы**

При запуске программы пользователь видит следующее окно:



Появилась возможность устанавливать время жизни объекта. При недопустимо введённом значении устанавливается значение по умолчанию



При нажатии на кнопку Show objects показывает все сгенерированные объекты (их классы, время рождения и ID).

**Листинг программы**

***Main.java***

public class Main {  
  
 public static void main(String[] args){  
  
 MainProgram program = new MainProgram();  
 program.run();  
 }  
}

***MainProgram.java***

import javax.swing.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.util.ArrayList;  
  
public class MainProgram extends JFrame {  
  
 /\*Panels\*/  
 static public JPanel *bees* = new JPanel();  
 static public JPanel *panel* = new JPanel();  
 /\*Window's size\*/  
 static public int *windowWIDTH* = 1100, *HEIGHT* = 600, *beesWIDTH* = 600, *panelWIDTH* = 500;  
 /\*Habitat\*/  
 Habitat habitat = new Habitat();  
 /\*Time\*/  
 private long timePrev = 0, startTime, timeStop = 0, timeContinue = 0;  
 private int time;  
 /\*Boolean variables\*/  
 private boolean bText = false;  
 private boolean bSimulation = false;  
 private boolean slWorker = true;  
 /\*Text fields\*/  
 private JLabel textTimeSimulation = new JLabel(""), textPeriod = new JLabel("Введите период появления Рабочих и Трутней");  
 private JTextArea textResults = new JTextArea("");  
 private JTextField intWorker = new JTextField(""), intDrone = new JTextField("");  
 private JLabel textLifeTime = new JLabel("Введите время жизни Рабочих и Трутней");  
 private JTextField workerLifeTime = new JTextField(""), droneLifeTime = new JTextField("");  
 /\*Buttons\*/  
 private JButton btStart = new JButton("Start"), btStop = new JButton("Stop"),  
 btOk = new JButton("Ok"), btCancel = new JButton("Cancel");  
 private JToggleButton btInfo = new JToggleButton("On/Off INFO");  
 private JRadioButton timeOn = new JRadioButton("Показать время симуляции"),  
 timeOff = new JRadioButton("Скрыть время симуляции");  
 private JButton btShowObjects = new JButton("Show objects");  
 /\*Sliders\*/  
 private String arrSliders[] = {"Процент появления рабочего", "'Процент' появления трутня"};  
 private JComboBox<String> comboBoxSliders = new JComboBox<String>(arrSliders);  
 private JSlider slBees = new JSlider(0, 100, 50);  
 /\*Menu\*/  
   
 private JMenu jmCommands = new JMenu("Комманды");  
 private JMenuItem jmiStart = new JMenuItem("Старт"), jmiStop = new JMenuItem("Стоп"), jmiTime = new JMenuItem("Показать/скрыть время симуляции");  
 private JToolBar jToolBar = new JToolBar();  
 private JButton jtbStart = new JButton("Старт"), jtbStop = new JButton("Стоп"), jtbTime = new JButton("Показать/скрыть время симуляции");  
  
 public void run(){  
 init();  
 event\_handling();  
 game\_loop();  
 }  
  
 private void game\_loop(){  
  
 while(true){  
 print\_time\_simulation();  
 if(bSimulation) {  
 long timeCur = System.*currentTimeMillis*() - startTime;  
 long step = timeCur - timePrev;  
 if (step >= 1000) {  
 time = (int) (timeCur / 1000);  
 habitat.update(time);  
 timePrev = timeCur;  
 }  
 }  
 }  
 }  
  
 private void init(){  
 this.setLayout(null);  
 this.setSize(*windowWIDTH*, *HEIGHT*);  
 this.setDefaultCloseOperation(this.*EXIT\_ON\_CLOSE*);  
 this.setVisible(true);  
 this.requestFocus();  
  
 *bees*.setLayout(null);  
 *bees*.setBounds(0,0,*beesWIDTH*, *HEIGHT*);  
  
 *panel*.setLayout(null);  
 *panel*.setBounds(*beesWIDTH*,0, *panelWIDTH*, *HEIGHT*);  
 textTimeSimulation.setBounds(0,0, 300, 20);  
 *panel*.add(textTimeSimulation);  
 btStart.setBounds(45, 30, 100, 50);  
 *panel*.add(btStart);  
 btStop.setBounds(155, 30, 100, 50);  
 *panel*.add(btStop);  
 timeOn.setBounds(0, 90, 300, 15);  
 *panel*.add(timeOn);  
 timeOff.setBounds(0, 115, 300, 15);  
 *panel*.add(timeOff);  
 ButtonGroup buttonGroup = new ButtonGroup();  
 buttonGroup.add(timeOn);  
 buttonGroup.add(timeOff);  
 btInfo.setBounds(100, 140, 100, 50);  
 *panel*.add(btInfo);  
 textResults.setBounds(0, 200, 300, 65);  
 *panel*.add(textResults);  
 btOk.setBounds(45, 275, 100, 50);  
 *panel*.add(btOk);  
 btCancel.setBounds(155, 275, 100, 50);  
 *panel*.add(btCancel);  
 textPeriod.setBounds(0, 335, 300, 15);  
 *panel*.add(textPeriod);  
 intWorker.setBounds(0, 360, 100, 15);  
 *panel*.add(intWorker);  
   
 *panel*.add(intDrone);  
 comboBoxSliders.setBounds(50, 385, 200, 20);  
 *panel*.add(comboBoxSliders);  
 slBees.setBounds(20, 415, 260, 50);  
 slBees.setMajorTickSpacing(10);  
 slBees.setPaintLabels(true);  
 slBees.setSnapToTicks(true);  
 *panel*.add(slBees);  
 textLifeTime.setBounds(0, 465, 300, 15);  
 *panel*.add(textLifeTime);  
 workerLifeTime.setBounds(0, 490, 100, 15);  
 *panel*.add(workerLifeTime);  
 droneLifeTime.setBounds(200, 490, 100, 15);  
 *panel*.add(droneLifeTime);  
 btShowObjects.setBounds(300, 140, 120, 50);  
 *panel*.add(btShowObjects);  
  
 jmCommands.add(jmiStart);  
 jmCommands.add(jmiStop);  
 jmCommands.add(jmiTime);  
 jMenuBar.add(jmCommands);  
  
 jToolBar.add(jtbStart);  
 jToolBar.add(jtbStop);  
 jToolBar.add(jtbTime);  
 jToolBar.setBounds(300, 25, 100, 100);  
 *panel*.add(jToolBar);  
  
 textResults.setEditable(false);  
 btStop.setEnabled(false);  
 btInfo.setEnabled(false);  
 btOk.setEnabled(false);  
 btCancel.setEnabled(false);  
  
 this.add(*bees*);  
 this.add(*panel*);  
 this.setJMenuBar(jMenuBar);  
 this.setResizable(false);  
 }  
  
 private void event\_handling(){  
 this.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 if(String.*valueOf*(e.getKeyChar()).equals("t") || String.*valueOf*(e.getKeyChar()).equals("T")){  
 change\_T();  
 }  
 if(String.*valueOf*(e.getKeyChar()).equals("b") || String.*valueOf*(e.getKeyChar()).equals("B")){  
 change\_B();  
 }  
 if(String.*valueOf*(e.getKeyChar()).equals("e") || String.*valueOf*(e.getKeyChar()).equals("E")){  
 change\_E();  
 }  
 }  
 });  
 btStart.addActionListener(e -> {  
 change\_B();  
 this.requestFocus();  
 });  
 btStop.addActionListener(e -> {  
 change\_E();  
 this.requestFocus();  
 });  
 btInfo.addActionListener(e -> {  
 this.requestFocus();  
 if(!bSimulation) {  
 if (btInfo.isSelected()) {  
 print\_results(true);  
 btOk.setEnabled(true);  
 btCancel.setEnabled(true);  
 } else {  
 print\_results(false);  
 btOk.setEnabled(false);  
 btCancel.setEnabled(false);  
 }  
 }  
 });  
 timeOn.addActionListener(e -> {  
 bText = true;  
 this.requestFocus();  
 });  
 timeOff.addActionListener(e -> {  
 bText = false;  
 this.requestFocus();  
 });  
 btOk.addActionListener(e -> {  
 this.requestFocus();  
 bSimulation = false;  
 btStart.setEnabled(true);  
 btStop.setEnabled(false);  
 jmiStart.setEnabled(true);  
 jmiStop.setEnabled(false);  
 jtbStart.setEnabled(true);  
 jtbStop.setEnabled(false);  
 btOk.setEnabled(false);  
 btCancel.setEnabled(false);  
 btInfo.setEnabled(false);  
 btInfo.setSelected(false);  
 timeStop = timeContinue = 0;  
 habitat.clear();  
 print\_results(false);  
 time = 0;  
 });  
 btCancel.addActionListener(e -> {  
 this.requestFocus();  
 timeContinue = System.*currentTimeMillis*();  
 startTime += timeContinue - timeStop;  
 bSimulation = true;  
 btStart.setEnabled(false);  
 btStop.setEnabled(true);  
 jmiStart.setEnabled(false);  
 jmiStop.setEnabled(true);  
 jtbStart.setEnabled(false);  
 jtbStop.setEnabled(true);  
 btOk.setEnabled(false);  
 btCancel.setEnabled(false);  
 btInfo.setEnabled(false);  
 btInfo.setSelected(false);  
 print\_results(false);  
 });  
 intWorker.addActionListener(e -> {  
 try {  
 int tmp = Integer.*parseInt*(intWorker.getText());  
 if(tmp > 0){  
 Habitat.*N2* = tmp;  
 JOptionPane.*showMessageDialog*(this, "Установлен период появления\n" +  
 "рабочих, равный " + Habitat.*N2* + " секунд.");  
 }else{  
 throw new Exception();  
  
 }  
 }catch (Exception e1){  
 Habitat.*N2* = 1;  
 JOptionPane.*showMessageDialog*(this, "Период не может быть отрицательным!" +  
 "\nУстановлено значение по умолчанию = 1.");  
 }  
 intWorker.setText("");  
 this.requestFocus();  
 });  
 intDrone.addActionListener(e -> {  
 try {  
 int tmp = Integer.*parseInt*(intDrone.getText());  
 if (tmp > 0) {  
 Habitat.*N1* = tmp;  
 JOptionPane.*showMessageDialog*(this, "Установлен период появления\n" +  
 "трутней, равный " + Habitat.*N1* + " секунд.");  
 } else {  
 throw new Exception();  
 }  
 }catch (Exception e1){  
 Habitat.*N1* = 1;  
 JOptionPane.*showMessageDialog*(this, "Период не может быть отрицательным!" +  
 "\nУстановлено значение по умолчанию = 1.");  
 }  
 intDrone.setText("");  
 this.requestFocus();  
 });  
 comboBoxSliders.addActionListener(e -> {  
 this.requestFocus();  
 String tmp = (String)comboBoxSliders.getSelectedItem();  
 if(tmp.equals("Процент появления рабочего")){  
 slWorker = true;  
 }else{  
 slWorker = false;  
 }  
 });  
 slBees.addChangeListener(e -> {  
 this.requestFocus();  
 if(slWorker){  
 Habitat.*P* = slBees.getValue();  
 }else{  
 Habitat.*K* = slBees.getValue();  
 }  
 });  
 jmiStart.addActionListener(e -> {  
 change\_B();  
 this.requestFocus();  
 });  
 jmiStop.addActionListener(e -> {  
 change\_E();  
 this.requestFocus();  
 });  
 jmiTime.addActionListener(e -> {  
 change\_T();  
 this.requestFocus();  
 });  
 jtbStart.addActionListener(e -> {  
 change\_B();  
 this.requestFocus();  
 });  
 jtbStop.addActionListener(e -> {  
 change\_E();  
 this.requestFocus();  
 });  
 jtbTime.addActionListener(e -> {  
 change\_T();  
 this.requestFocus();  
 });  
 workerLifeTime.addActionListener(e -> {  
 try {  
 int tmp = Integer.*parseInt*(workerLifeTime.getText());  
 if(tmp > 0){  
 Habitat.*workerLifeTime* = tmp;  
 JOptionPane.*showMessageDialog*(this, "Установлено время жизни\n" +  
 "рабочих, равное " + Habitat.*workerLifeTime* + " секунд.");  
 }else{  
 throw new Exception();  
  
 }  
 }catch (Exception e1){  
 Habitat.*workerLifeTime* = 5;  
 JOptionPane.*showMessageDialog*(this, "Время жизни не может быть отрицательным!" +  
 "\nУстановлено значение по умолчанию = 5.");  
 }  
 workerLifeTime.setText("");  
 this.requestFocus();  
 });  
 droneLifeTime.addActionListener(e -> {  
 try {  
 int tmp = Integer.*parseInt*(droneLifeTime.getText());  
 if (tmp > 0) {  
 Habitat.*droneLifeTime* = tmp;  
 JOptionPane.*showMessageDialog*(this, "Установлено время жизни\n" +  
 "трутней, равное " + Habitat.*droneLifeTime* + " секунд.");  
 } else {  
 throw new Exception();  
 }  
 }catch (Exception e1){  
 Habitat.*droneLifeTime* = 5;  
 JOptionPane.*showMessageDialog*(this, "Время жизни не может быть отрицательным!" +  
 "\nУстановлено значение по умолчанию = 5.");  
 }  
 droneLifeTime.setText("");  
 this.requestFocus();  
 });  
 btShowObjects.addActionListener(e -> {  
 String obj[] = new String[habitat.getArrBees().size() + 1];  
 int count = 0;  
 obj[count++] = "Класс Время рождения ID";  
 for(Character bee : habitat.getArrBees()){  
 if(bee instanceof Worker){  
 obj[count++] = "Рабочий " + Integer.*toString*(bee.getBornTime()) + " " + Integer.*toString*(bee.getId());  
 }  
 else{  
 obj[count++] = "Трутень " + Integer.*toString*(bee.getBornTime()) + " " + Integer.*toString*(bee.getId());  
 }  
 }  
 JList<String> jlst = new JList<String>(obj);  
 JScrollPane jscrl = new JScrollPane(jlst);  
 JOptionPane.*showMessageDialog*(this, jscrl);  
  
 this.requestFocus();  
 });  
 }  
  
 private void change\_T(){  
 bText = !bText;  
 }  
  
 private void change\_B(){  
 startTime = System.*currentTimeMillis*();  
 bSimulation = true;  
 btStart.setEnabled(false);  
 btStop.setEnabled(true);  
 jmiStart.setEnabled(false);  
 jmiStop.setEnabled(true);  
 jtbStart.setEnabled(false);  
 jtbStop.setEnabled(true);  
 btOk.setEnabled(false);  
 btCancel.setEnabled(false);  
 btInfo.setEnabled(false);  
 Worker.*beeWorker* = Drone.*beeDrone* = 0;  
 habitat.clear();  
 timePrev = 0;  
 }  
  
   
  
 private void print\_time\_simulation(){  
 if(bText){  
 textTimeSimulation.setText(time+" секунд после начала симуляции");  
 }else {  
 textTimeSimulation.setText("");  
 }  
 *panel*.repaint();  
 }  
  
 private void print\_results(boolean TF){  
 if(TF) {  
 textResults.setText("Количество сгенерированных пчёл-рабочих: " + Worker.*beeWorker* +  
 "\nКоличество сгенерированных пчёл-трутней: " + Drone.*beeDrone* +  
 "\nОбщее количество сгенерированных пчёл: " + (Drone.*beeDrone* + Worker.*beeWorker*) +  
 "\nВремя симуляции: " + time);  
 }else{  
 textResults.setText("");  
 }  
 }  
}

***Habitat.java***

import javax.swing.\*;  
import java.awt.\*;  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.TreeMap;  
  
public class Habitat {  
  
 private ArrayList<Character> arrBees = new ArrayList<Character>();  
 private HashSet<Integer> arrId = new HashSet<Integer>();  
 private TreeMap<Integer, Integer> arrBornTime = new TreeMap<Integer, Integer>();  
 private int WIDTH = MainProgram.*beesWIDTH*, HEIGHT = MainProgram.*HEIGHT*;  
 static public int *N1* = 3, *N2* = 2, *K* = 40, *P* = 80, *workerLifeTime* = 3, *droneLifeTime* = 3;  
 private int imageSize = 50;  
 private ImageIcon imageIcon = new ImageIcon(new ImageIcon(getClass().getResource(  
 "/Images/sky2.png")).getImage().getScaledInstance(WIDTH,HEIGHT, Image.*SCALE\_DEFAULT*));  
  
 Habitat(){  
 JLabel tmp = new JLabel();  
 tmp.setBounds( 0, 0, imageIcon.getIconWidth(), imageIcon.getIconHeight());  
 tmp.setIcon(imageIcon);  
 MainProgram.*bees*.add(tmp, 0, 0);  
 MainProgram.*bees*.repaint();  
 }  
 void update(int time){  
 removeBees(time);  
  
 if(time % *N2* == 0)  
 {  
 int random = (int)Math.*floor*(Math.*random*()\*100);  
 if(random <= *P*) {  
 int x = (int) Math.*floor*(Math.*random*() \* (WIDTH - imageSize));  
 int y = (int) Math.*floor*(Math.*random*() \* (HEIGHT - imageSize));  
 Worker worker = new Worker(x, y, *workerLifeTime*, time, arrId);  
 arrBees.add(worker);  
 arrId.add(worker.getId());  
 arrBornTime.put(worker.getId(), worker.getBornTime());  
 Worker.*beeWorker*++;  
 }  
 }  
 if(time % *N1* == 0 && Worker.*beeWorker* != 0){  
 int dronePercent = 100\*Drone.*beeDrone*/(Drone.*beeDrone* + Worker.*beeWorker*);  
 if(dronePercent < *K*){  
 int x = (int)Math.*floor*(Math.*random*()\*(WIDTH-imageSize));  
 int y = (int)Math.*floor*(Math.*random*()\*(HEIGHT-imageSize));  
 Drone drone = new Drone(x, y, *droneLifeTime*, time, arrId);  
 arrBees.add(drone);  
 arrId.add(drone.getId());  
 arrBornTime.put(drone.getId(), drone.getBornTime());  
 Drone.*beeDrone*++;  
 }  
 }  
 }  
 void clear(){  
 MainProgram.*bees*.removeAll();  
 new Habitat();  
 }  
 void removeBees(int time){  
 Iterator<Character> iterator = arrBees.iterator();  
 while (iterator.hasNext()){  
 Character bee = iterator.next();  
 if((time - bee.getBornTime()) >= bee.getLifeTime()){  
 MainProgram.*bees*.remove(bee.getJlb());  
 MainProgram.*bees*.repaint();  
 if(bee instanceof Worker){  
 Worker.*beeWorker*--;  
 }  
 else{  
 Drone.*beeDrone*--;  
 }  
 arrId.remove(bee.getId());  
 arrBornTime.remove(bee.getId());  
 iterator.remove();  
 }  
 }  
 }  
 ArrayList<Character> getArrBees(){return arrBees;}  
}

***Worker.java***

import javax.swing.\*;  
import java.awt.\*;  
import java.util.HashSet;  
  
public class Worker extends Character{  
  
 static public int *beeWorker* = 0;  
 ImageIcon imageIcon = new ImageIcon(new ImageIcon(getClass().getResource(  
 "/Images/Worker.png")).getImage().getScaledInstance(50,50,Image.*SCALE\_DEFAULT*));  
  
  
 public Worker(int x, int y, int lifeTime, int bornTime, HashSet<Integer> arr){  
 JLabel jlb = new JLabel();  
 jlb.setBounds(x, y, imageIcon.getIconWidth(), imageIcon.getIconHeight());  
 jlb.setIcon(imageIcon);  
 setJlb(jlb);  
 MainProgram.*bees*.add(getJlb(),0,0);  
 MainProgram.*bees*.repaint();  
 setx(x);  
 sety(y);  
 setLifeTime(lifeTime);  
 setBornTime(bornTime);  
 setId(arr);  
 }  
}

***Drone.java***

import javax.swing.\*;  
import java.awt.\*;  
import java.util.HashSet;  
  
public class Drone extends Character {  
  
 static public int *beeDrone* = 0;  
 ImageIcon imageIcon = new ImageIcon(new ImageIcon(getClass().getResource(  
 "/Images/Drone.png")).getImage().getScaledInstance(50,50,Image.*SCALE\_DEFAULT*));  
  
 public Drone(int x, int y, int lifeTime, int bornTime, HashSet<Integer> arr){  
 JLabel jlb = new JLabel();  
 jlb.setBounds(x, y, imageIcon.getIconWidth(), imageIcon.getIconHeight());  
 jlb.setIcon(imageIcon);  
 setJlb(jlb);  
 MainProgram.*bees*.add(getJlb(),0,0);  
 MainProgram.*bees*.repaint();  
 setx(x);  
 sety(y);  
 setLifeTime(lifeTime);  
 setBornTime(bornTime);  
 setId(arr);  
 }  
}

***Character.java***

import javax.swing.\*;  
import java.util.HashSet;  
import java.util.Random;  
  
public abstract class Character implements iBehaviour {  
  
 private int x, y;  
 private JLabel jlb;  
 private int lifeTime, bornTime;  
 private int id;  
  
 @Override  
 public void movexy(int x, int y) {}  
 @Override  
 public void setx(int x) {this.x = x;}  
   
 @Override  
 public int getx() {return this.x;}  
 @Override  
 public int gety() {return this.y;}  
 @Override  
 public int getLifeTime(){return this.lifeTime;}  
 @Override  
 public int getBornTime(){return this.bornTime;}  
 @Override  
 public void setLifeTime(int lifeTime){this.lifeTime = lifeTime;}  
 @Override  
 public void setBornTime(int bornTime){this.bornTime = bornTime;}  
 @Override  
 public int getId(){return id;}  
 @Override  
 public void setId(HashSet<Integer> arr){  
 while(true) {  
 int tmp = new Random().nextInt();  
 if(tmp < 0) tmp = -tmp;  
 if(!arr.contains(tmp)){  
 id = tmp;  
 break;  
 }  
 }  
 }  
 @Override  
 public JLabel getJlb(){return jlb;}  
 @Override  
 public void setJlb(JLabel jlb){this.jlb = jlb;}  
}

***iBehaviour.java***

import javax.swing.\*;  
import java.util.HashSet;  
  
public interface iBehaviour {  
 void movexy(int x, int y);  
 void setx(int x);  
 void sety(int y);  
 int getx();  
 int gety();  
 int getLifeTime();  
 void setLifeTime(int lifeTime);  
 int getBornTime();  
 void setBornTime(int bornTime);  
 int getId();  
 void setId(HashSet<Integer> arr);  
 JLabel getJlb();  
 void setJlb(JLabel jlb);  
}