**Лабораторная работа №6**

Выполнил

студент І курса

группа ИВ-32

Гутов В.В.

Номер ЗК: 3210

Номер ЗК: 3210

С13=12

Підготувати ігрову кімнату для дітей різних вікових груп. Іграшок повинно бути фіксована кількість у межах виділеної суми грошей. Повинні зустрічатися іграшки для різних вікових груп: маленькі, середні та великі машинки, ляльки, м'ячі, кубики. Провести сортування іграшок у кімнаті за будь-яким параметром. Знайти іграшку в кімнаті, що відповідає заданому діапазону вартості.

Код программы

package OOP.lab6;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Random;

/\*\*

\*

\* @version 1 29 April 2014

\* @author Gutov Vladislav

\*

\*/

public class Main {

public static void main(String[] args){

ArrayList<Toy> toys = new ArrayList<Toy>();

int cost = 400;

int i = 0;

do{

Toy currentToy; //temp variable

Random rand = new Random();

int random = rand.nextInt(3);

//random chose of which object to be added to list

if(random == 0)

currentToy = new Ball(rand.nextInt(30) + 20, i % 12 + 1, i % 3 + 1, "blue");

else if(random == 1)

currentToy = new Doll(rand.nextInt(30) + 20, i % 12 + 1, rand.nextBoolean());

else

currentToy = new Car(rand.nextInt(30) + 20, i % 12 + 1, rand.nextBoolean());

i+=currentToy.getCost();

toys.add(currentToy);

}while (i < cost);

//printing list on screen

for(Toy toy : toys){

System.out.println(toy);

}

System.out.println();

System.out.println("i = " + i);

System.out.println();

//sorting

Collections.sort(toys, Toy.comparator);

for(Toy toy : toys){

System.out.println(toy);

}

System.out.println();

//searching

ArrayList<Toy> findedToys = Toy.search(toys, 30, 40);

for(Toy toy : findedToys){

System.out.println(toy);

}

}

}

package OOP.lab6;

import java.util.ArrayList;

import java.util.Comparator;

/\*\*

\*

\* @version 1 29 April 2014

\* @author Gutov Vladislav

\*

\* Abstract class Toy was created for inherit from him another classes

\*/

public abstract class Toy {

private int cost;

private int age;

public static CompareByCost comparator = new CompareByCost(); //comparator for sorting all inherited classes

/\*\*

\* Constructor with 2 field: cost, age

\* @param cost - toy price

\* @param age - sets in special borders

\*/

public Toy(int cost, int age){

this.cost = cost;

setAge(age);

}

/\*\*

\*

\* Search all elements by cost in specified borders

\*

\* @param toys - ArrayList in where should search elements

\* @param key1 - left border(exclude)

\* @param key2 - right border(exclude)

\* @return ArrayList with found elements

\*/

public static ArrayList<Toy> search(ArrayList<Toy> toys, int key1, int key2){

ArrayList<Toy> result = new ArrayList<Toy>();

for(Toy toy : toys)

if(toy.cost > key1 && toy.cost < key2)

result.add(toy);

return result;

}

public int getCost() {return cost;}

public int getAge() {return age;}

public void setCost(int cost) {this.cost = cost;}

/\*\*

\* Sets age in special borders

\* @param age - age

\*/

public void setAge(int age) {

this.age = 99;

if(age <= 12)

this.age = 12;

if(age <= 8)

this.age = 8;

if(age <= 6)

this.age = 6;

if (age < 5)

this.age = 3;

}

/\*\*

\* Compare two inherited classes from Toy by cost

\*/

private static class CompareByCost implements Comparator<Toy>{

@Override

public int compare(Toy o1, Toy o2) {

return o2.getCost() - o1.getCost();

}

}

}

package OOP.lab6;

/\*\*

\*

\* @version 1 29 April 2014

\* @author Gutov Vladislav

\*

\* Class Ball

\*/

public class Ball extends Toy{

//size constants

public static final int SMALL = 1;

public static final int MEDIUM = 2;

public static final int LARGE = 3;

private int size;

private String color;

public Ball(int cost, int age, int size, String color){

super(cost, age);

this.size = size;

this.color = new String(color);

}

public int getSize() {return size;}

public void setSize(int size) {this.size = size;}

public String getColor() {return new String(color);}

public void setColor(String color) {this.color = new String(color);}

//was overridden for future needs in 7 lab.work

@Override

public boolean equals(Object obj) {

if(obj == this)

return true;

if(obj == null)

return false;

if(!(obj instanceof Ball))

return false;

Ball tObj = (Ball) obj;

if(tObj.getAge() == getAge() && tObj.getCost() == getCost() && tObj.getColor().equals(getColor()) && tObj.getSize() == getSize())

return true;

return false;

}

@Override

public String toString(){

return "Ball = {\"cost = " + getCost() + "\", \"age = " + getAge()+ "\", \"size = " + size + "\", \"color = " + color + "\"}";

}

}

package OOP.lab6;

/\*\*

\* This class very similar to {@link OOP.lab6.Ball Ball}

\*/

public class Car extends Toy {

private boolean mechanical;

public Car(int cost, int age, boolean mechanical){

super(cost, age);

this.mechanical = mechanical;

}

public boolean isMechanical() {return mechanical;}

public void setMechanical(boolean mechanical) {this.mechanical = mechanical;}

@Override

public boolean equals(Object obj) {

if(obj == this)

return true;

if(obj == null)

return false;

if(!(obj instanceof Car))

return false;

Car tObj = (Car) obj;

if(tObj.getAge() == getAge() && tObj.getCost() == getCost() && tObj.mechanical == mechanical)

return true;

return false;

}

@Override

public String toString(){

return "Car = {\"cost = " + getCost()+ "\", \"age = " + getAge() + "\", \"mechanical = " + mechanical + "\"}";

}

}

package OOP.lab6;

/\*\*

\* This class very similar to {@link OOP.lab6.Ball Ball}

\*/

public class Doll extends Toy{

private boolean sex;

public Doll(int cost, int age, boolean sex){

super(cost, age);

this.sex = sex;

}

public boolean isSex() {return sex;}

public void setSex(boolean sex) {this.sex = sex;}

public boolean equals(Object obj) {

if(obj == this)

return true;

if(obj == null)

return false;

if(!(obj instanceof Car))

return false;

Doll tObj = (Doll) obj;

if(tObj.getAge() == getAge() && tObj.getCost() == getCost() && tObj.sex == sex)

return true;

return false;

}

@Override

public String toString(){

return "Doll = {\"cost = " + getCost()+ "\", \"age = " + getAge() + "\", \"sex = " + sex + "\"}";

}

}

Выводы

В данной лабораторной работе я более детально ознакомился с принципами объектно-ориентированного программирования: изучил механизм наследования классов и полиморфизм. В программе подготавливается игровая комната для детей разных возрастных групп на указанную цену. Затем все игрушки сортируются по цене, а после происходит поиск игрушек в указанном диапазоне цен.