

Факультет программной инженерии и компьютерной техники

Дисциплина: Программирование

Лабораторная работа №4

Вариант № 311584,4

Выполнил: Васильев А.Ю.

№ группы: P3115

Преподаватель: Письмак А.Е.

2020 год, 1 семестр

1.Текст задания.

**Описание предметной области, по которой должна быть построена объектная модель:**

Малыш горячо обнял ее. Потом они пропели Малышу "Многие лета", а Малыш выстрелил, и треск получился оглушительный. Все утро Малыш то и дело стрелял из пистолета, ждал гостей и все время размышлял о словах папы, что подарки могут появиться и днем. На какой-то счастливый миг он вдруг поверил, что свершится чудо -- ему подарят собаку. Но тут же понял, что это невозможно, и даже рассердился на себя за то, что так глупо размечтался. Ведь он твердо решил не думать сегодня о собаке и всему радоваться. И Малыш действительно всему радовался. Сразу же после обеда мама стала накрывать на стол у него в комнате. Она поставила в вазу большой букет цветов и принесла самые красивые розовые чашки. Три штуки. Малыш замялся. Теперь ему надо было рассказать, что он пригласил на день рождения Карлсона, хотя мама, конечно, будет этим недовольна.

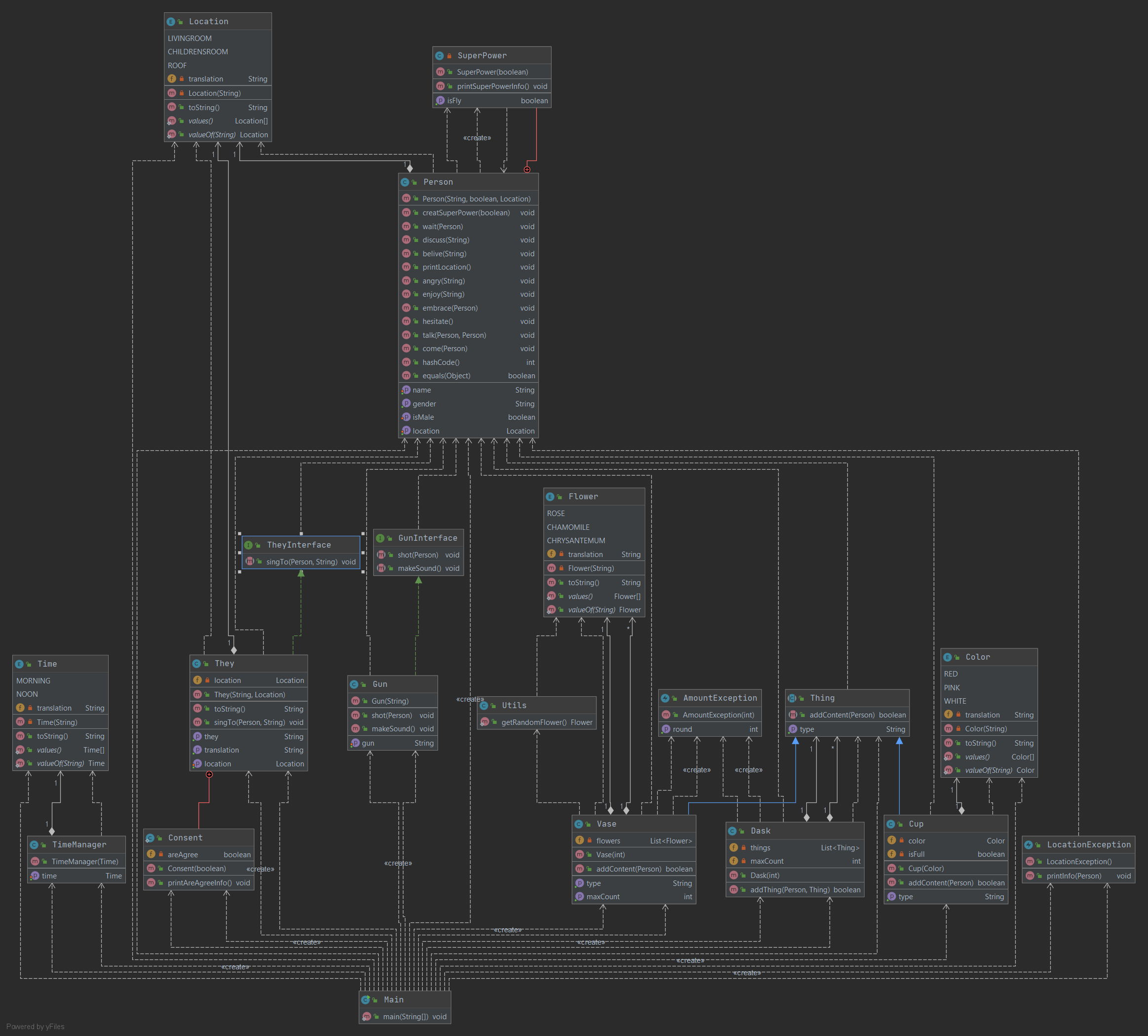
**Программа должна удовлетворять следующим требованиям:**

1. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
2. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

**Порядок выполнения работы:**

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

2. Диаграмма классов объектной модель.



3.Исходный код.

Главный класс.

package prog;  
public class Main {  
 public static void main(String[] args) {  
 TimeManager manager = new TimeManager(Time.*MORNING*);  
 Person child = new Person("Малыш", true, Location.*LIVINGROOM*);  
 Person mom = new Person("Мама", false, Location.*LIVINGROOM*);  
 Person guest1 = new Person("гость Лёня", true, Location.*LIVINGROOM*);  
 Person guest2 = new Person("гость Маша", false, Location.*LIVINGROOM*);  
 Person guest3 = new Person("гость Саша", true, Location.*LIVINGROOM*);  
 Person guest4 = new Person("гость Карласон", true, Location.*ROOF*) {  
 @Override  
 public void come(Person who) {  
 printLocation();  
 setLocation(Location.*CHILDRENSROOM*);  
 System.*out*.println(getName() + " прилетел к " + who.getName() + " на праздник");  
 printLocation();  
 }  
 };  
 System.*out*.println(manager.getTime());  
 guest4.creatSuperPower(true);  
 They they = new They("Они", Location.*LIVINGROOM*);  
 child.printLocation();  
 child.embrace(mom);  
 They.Consent consent = new They.Consent(true);  
 consent.printAreAgreeInfo();  
 they.getLocation();  
 they.singTo(child, "\"Многие лета\"");  
 Gun gun = new Gun("пистолет");  
 gun.shot(child);  
 child.wait(guest1);  
 child.wait(guest2);  
 child.wait(guest3);  
 child.discuss("словах папы, что подарки могут появиться и днем");  
 child.belive("свершится чудо - ему подарят собаку");  
 child.angry("размечтался");  
 child.enjoy("не думал о собаке");  
 child.setLocation(Location.*CHILDRENSROOM*);  
 manager.setTime(Time.*NOON*);  
 mom.setLocation(Location.*CHILDRENSROOM*);  
 System.*out*.println(manager.getTime());  
 child.printLocation();  
 Vase vase = new Vase(-11);  
 Dask dask = new Dask(-4);  
 try {  
 if (mom.getLocation() != child.getLocation()) throw new LocationException();  
 for (int i = 0; i < vase.getMaxCount(); i++) {  
 vase.addContent(mom);  
 }  
 dask.addThing(mom, vase);  
 for (int i = 0; i < 3; i++) {  
 Thing temp = new Cup(Color.*PINK*);  
 temp.addContent(mom);  
 dask.addThing(mom, temp);  
 }  
 }catch (LocationException e) {  
 e.printInfo(child);  
 }  
 guest1.setLocation(Location.*CHILDRENSROOM*);  
 guest3.setLocation(Location.*CHILDRENSROOM*);  
 guest1.come(child);  
 guest3.come(child);  
 child.hesitate();  
 child.talk(guest4, mom);  
 guest4.come(child);  
 }  
}

Класс Utils.

package prog;  
  
public class Utils {  
  
 public static Flower getRandomFlower() {  
 int random = 1 + (int) (Math.*random*() \* 4);  
 if(random == 1){  
 return Flower.*ROSE*;  
 }  
 else if(random == 2){  
 return Flower.*CHAMOMILE*;  
 }  
 else{  
 return Flower.*CHRYSANTEMUM*;  
 }  
 }  
  
}

Класс TimeManager.

package prog;  
  
public class TimeManager{  
 private Time time;  
 public TimeManager(Time time){  
 this.time = time;  
 }  
  
 public void setTime(Time time) {  
 this.time = time;  
 }  
  
 public Time getTime() {  
 return time;  
 }  
  
}

Enum Time.

package prog;  
public enum Time{  
 *MORNING*("утро"), *NOON*("полдень");  
 private String translation;  
  
 Time(String translation){  
 this.translation = translation;  
 }  
  
 @Override  
 public String toString() {  
 return translation;  
 }  
}

Enum Location.

package prog;  
public enum Location {  
 *LIVINGROOM*("гостинная"), *CHILDRENSROOM*("детская"), *ROOF*("крыша");  
 private String translation;  
  
 Location(String translation){  
 this.translation = translation;  
 }  
  
 @Override  
 public String toString() {  
 return translation;  
 }  
  
  
}

Класс Person.

package prog;  
import java.util.Objects;  
  
public class Person {  
 private String name;  
 private boolean isMale;  
 private Location location;  
  
 public Person(String name, boolean isMale, Location location){  
 this.name = name;  
 this.isMale = isMale;  
 this.location = location;  
 }  
  
 private class SuperPower{  
 private boolean isFly = false;  
 public SuperPower(boolean isFly){  
 this.isFly = isFly;  
 }  
 public boolean getIsFly() {  
 return isFly;  
 }  
 public void printSuperPowerInfo() {  
 if (getIsFly()) {  
 System.*out*.println(getName() + " умеет летать");  
 } else {  
 System.*out*.println(getName() + " имеет скрытые способности");  
 }  
 }  
 }  
  
 public void creatSuperPower(boolean isFly){  
 SuperPower power = new SuperPower(isFly);  
 power.printSuperPowerInfo();  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
  
 public void wait(Person person){  
 System.*out*.println(getName() + " ждал " + person.getName());  
 }  
  
 public void discuss(String about){  
 System.*out*.println(getName() + " размышлял о " + about);  
 }  
  
 public void belive(String about){  
 System.*out*.println(getName() + " поверил в то, что " + about);  
 }  
  
 public void setLocation(Location location) {  
 this.location = location;  
 }  
  
 public Location getLocation() {  
 return location;  
 }  
  
 public void printLocation(){  
 System.*out*.println(getName() + " находится в " + location);  
 }  
  
 public void angry(String about){  
 System.*out*.println(getName()+" рассердился на себя за то, что "+about);  
 }  
 public void enjoy(String about){  
 System.*out*.println(getName()+" радовался, потому что "+about);  
 }  
  
 public void setIsMale(boolean isMale) {  
 this.isMale = isMale;  
 }  
 public String getGender() {  
 if(isMale == true){  
 return "мужчина";  
 }else{  
 return "женщина";  
 }  
 }  
  
 public void embrace(Person who){  
 System.*out*.println(getName() + " горячо обнял " + who.getName());  
 }  
  
 public void hesitate(){  
 System.*out*.println(getName() + " замялся");  
 }  
  
 public void talk(Person who, Person whoNeed){  
 System.*out*.println(getName() + " должен рассказать " + whoNeed.getName() +", что на день рождение он пригласил " + who.getName());  
 }  
  
 public void come(Person who){  
 getLocation();  
 System.*out*.println(getName() + " пришел к " + who.getName());  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getName(), getGender(), getLocation());  
 }  
 @Override  
 public boolean equals(Object object){  
 if (object == null || object.getClass() != getClass()) {  
 return false;  
 }  
 else{  
 Person temp = (Person) object;  
 return Objects.*equals*(name, temp.name);  
 }  
 }  
}

Класс They.

package prog;  
  
public class They implements TheyInterface{  
 private String they;  
 private Location location;  
 public They(String they, Location location){  
 this.they=they;  
 this.location = location;  
 }  
  
 public static class Consent{  
 private boolean areAgree;  
 public Consent(boolean areAgree){  
 this.areAgree = areAgree;  
 }  
 public void printAreAgreeInfo(){  
 if (areAgree) {  
 System.*out*.println("Все с этим согласились");  
 } else {  
 System.*out*.println("Никто не обратил на это внимания");  
 }  
 }  
 }  
  
 public String getTranslation(){  
 return they;  
 }  
 @Override  
 public String toString(){  
 return "Перевод на русский: " + they;  
 }  
 @Override  
 public void singTo(Person person, String what) {  
 System.*out*.println(getTranslation() + " спели " + person.getName() + " " + what);  
 }  
 public void setLocation(Location location) {  
 this.location = location;  
 }  
  
 public Location getLocation() {  
 System.*out*.println(getThey() + " находится в " + location);  
 return location;  
 }  
  
 public String getThey() {  
 return they;  
 }  
}

Интерфейс TheyInterface.

package prog;  
  
public interface TheyInterface {  
 public void singTo(Person person, String what);  
}

Класс Gun.

package prog;  
  
public class Gun implements GunInterface {  
 private String gun;  
 public Gun(String gun){  
 this.gun = gun;  
 }  
  
 public void setGun(String gun){  
 this.gun = gun;  
 }  
  
 public String getGun() {  
 return gun;  
 }  
 @Override  
 public void shot(Person person){  
 System.*out*.print(person.getName() + " выстрелил из " + getGun() + ", ");  
 makeSound();  
 }  
 @Override  
 public void makeSound() {  
 System.*out*.println("c оглушительным треском");  
 }  
  
}

Интерфейс GunInterface.

package prog;  
  
public interface GunInterface {  
 public void shot(Person person);  
 public void makeSound();  
}

Класс Dask.

package prog;  
import java.util.ArrayList;  
import java.util.List;  
public class Dask {  
 private List<Thing> things = new ArrayList<Thing>();  
 private int maxCount;  
 public Dask(int maxCount){  
 try{  
 if (maxCount <= 0) throw new AmountException(maxCount);  
 this.maxCount = maxCount;  
 }catch (AmountException e){  
 this.maxCount = e.getRound();  
 }  
 }  
 public boolean addThing(Person person,Thing thing){  
 if(things.size() < maxCount) {  
 System.*out*.println(person.getName() + " добавила на стол " + thing.getType());  
 things.add(thing);  
 return true;  
 }else{  
 return false;  
 }  
 }  
}

Абстрактный класс Thing.

package prog;  
  
public abstract class Thing{  
 public abstract boolean addContent(Person person);  
 public abstract String getType();  
}

Класс Vase.

package prog;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Vase extends Thing {  
 private int maxCount;  
 private List<Flower> flowers = new ArrayList<Flower>();  
 public Vase(int maxFlowers){  
 try{  
 if (maxFlowers <= 0) throw new AmountException(maxFlowers);  
 maxCount = maxFlowers;  
 }catch (AmountException e){  
 maxCount=e.getRound();  
 }  
 }  
 public int getMaxCount() {  
 return maxCount;  
 }  
 @Override  
 public boolean addContent(Person person){  
 if(flowers.size() < maxCount) {  
 Flower temp = Utils.*getRandomFlower*();  
 flowers.add(temp);  
 System.*out*.println(person.getName()+" добваила " + temp);  
 return true;  
 }else{  
 System.*out*.println("Ваза полная");  
 return false;  
 }  
 }  
  
 @Override  
 public String getType() {  
 return "ваза";  
 }  
}

Enum Flower.

package prog;  
  
public enum Flower{  
 *ROSE*("роза"),*CHAMOMILE*("ромашка"),*CHRYSANTEMUM*("хризантема");  
 private String translation;  
  
 Flower(String translation){  
 this.translation = translation;  
 }  
  
 @Override  
 public String toString() {  
 return translation;  
 }  
}

Класс Cup.

package prog;  
  
public class Cup extends Thing {  
 private Color color;  
 private boolean isFull;  
 public Cup(Color color){  
 this.color = color;  
 }  
 @Override  
 public boolean addContent(Person person){  
 if(!isFull) {  
 System.*out*.println(person.getName() + " налила чай в кружку.");  
 isFull = true;  
 return true;  
 }else{  
 System.*out*.println(person.getName() + " не может налить чай в кружку, потому что она полная.");  
 return false;  
 }  
 }  
 @Override  
 public String getType() {  
 return color + " кружка";  
 }  
}

Enum Color.

package prog;  
  
public enum Color{  
 *RED*("красную"),*PINK*("розовую"),*WHITE*("белую");  
 private String translation;  
  
 Color(String translation){  
 this.translation = translation;  
 }  
  
 @Override  
 public String toString() {  
 return translation;  
 }  
}

Класс AmountException.

package prog;  
  
public class AmountException extends RuntimeException{  
 private int round;  
  
 public AmountException(int amount){  
 System.*out*.println("Нельзя поместить отрицательное число предметов, " + amount + "!");  
 round = Math.*abs*(amount);  
 }  
  
 public int getRound() {  
 return round;  
 }  
}

Класс LocationException.

package prog;  
  
public class LocationException extends Exception {  
 public LocationException(){  
 System.*out*.println("Нельзя поместить предметы на стол");  
 }  
 public void printInfo(Person person){  
 System.*out*.println("Мама должна переместиться к " + person.getName() + " и гостям в " + person.getLocation());  
 }  
}

4.Программа выводит.

утро

гость Карласон умеет летать

Малыш находится в гостинная

Малыш горячо обнял Мама

Все с этим согласились

Они находится в гостинная

Они спели Малыш "Многие лета"

Малыш выстрелил из пистолет, c оглушительным треском

Малыш ждал гость Лёня

Малыш ждал гость Маша

Малыш ждал гость Саша

Малыш размышлял о словах папы, что подарки могут появиться и днем

Малыш поверил в то, что свершится чудо - ему подарят собаку

Малыш рассердился на себя за то, что размечтался

Малыш радовался, потому что не думал о собаке

полдень

Малыш находится в детская

Нельзя поместить отрицательное число предметов, -11!

Нельзя поместить отрицательное число предметов, -4!

Мама добваила хризантема

Мама добваила хризантема

Мама добваила роза

Мама добваила ромашка

Мама добваила роза

Мама добваила роза

Мама добваила ромашка

Мама добваила хризантема

Мама добваила хризантема

Мама добваила роза

Мама добваила роза

Мама добавила на стол ваза

Мама налила чай в кружку.

Мама добавила на стол розовую кружка

Мама налила чай в кружку.

Мама добавила на стол розовую кружка

Мама налила чай в кружку.

Мама добавила на стол розовую кружка

гость Лёня пришел к Малыш

гость Саша пришел к Малыш

Малыш замялся

Малыш должен рассказать Мама, что на день рождение он пригласил гость Карласон

гость Карласон находится в крыша

гость Карласон прилетел к Малыш на праздник

гость Карласон находится в детская

5. Вывод.

В ходе данной лабораторной работы, была дополнена объектная модель классов. Также реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов. Еще добавлены локальный, анонимный и вложенный классы (static и non-static).