Факультет программной инженерии

И

Компьютерных технологий

Лабораторная работа №3

Вариант 494

Преподаватель: Письмак А.Е.

Выполнил: Патутин Владимир

Группа:P3114

Санкт-Петербург

2019

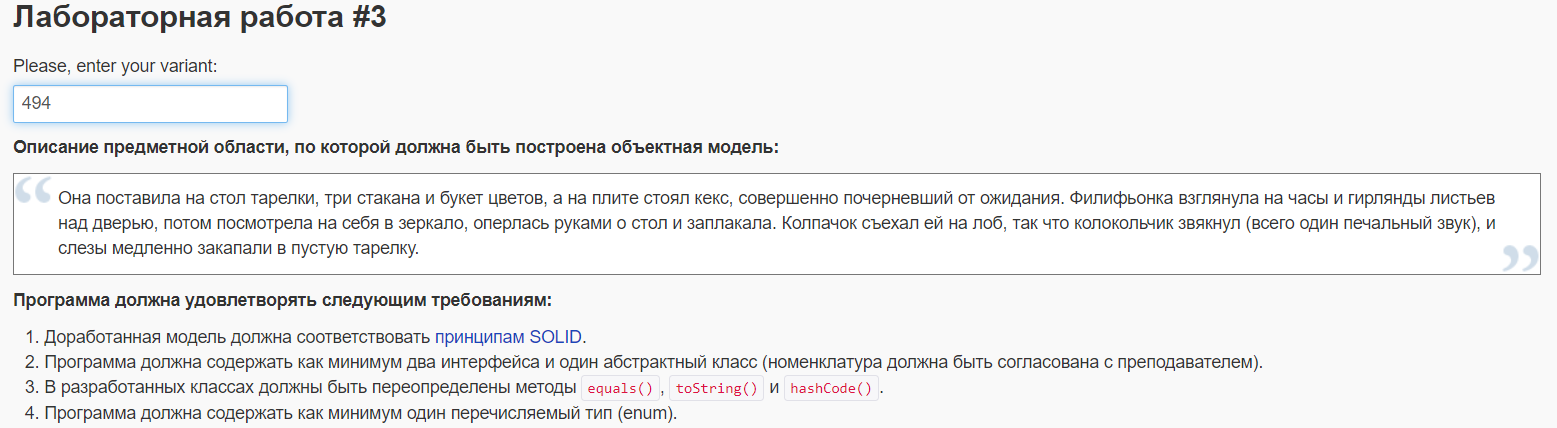
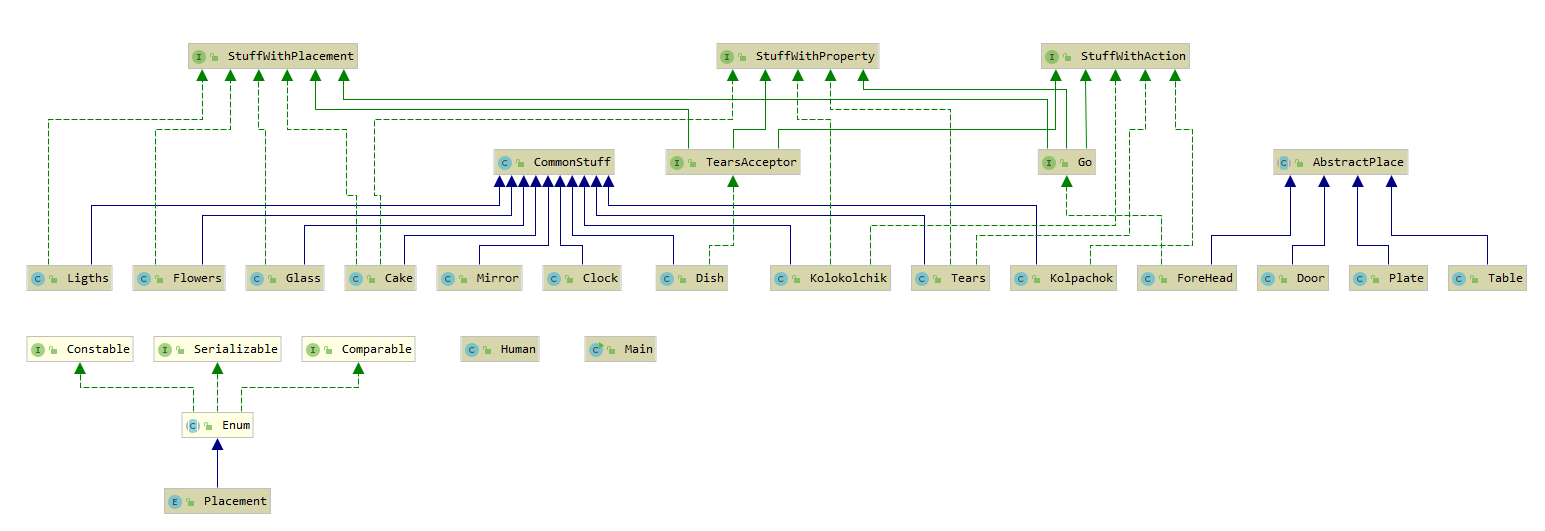


Диаграмма классов:



Исходный кот программы:

Main:

**package** Laba3;  
  
**import** Laba3.places.\*;  
**import** Laba3.stuff.\*;  
  
*/\*Она поставила на стол тарелки, три стакана и букет цветов, а на плите стоял кекс, совершенно почерневший от ожидания.  
 Филифьонка взглянула на часы и гирлянды листьев над дверью, потом посмотрела на себя в зеркало, оперлась руками о стол и заплакала.  
"Колпачок съехал ей на лоб", так что колокольчик звякнул (всего один печальный звук), и слезы медленно закапали в пустую тарелку.\*/***public class** Main {  
 **public static void** main(String[] args) {  
 Human Philiphonka = **new** Human(**"Филифьонка"**);  
 Philiphonka.aLive();  
 Table table = **new** Table();  
 Plate plate = **new** Plate();  
 Door door = **new** Door();  
 ForeHead foreHead = **new** ForeHead();  
 Dish dish = **new** Dish(table.getPlacement(), table);  
 Glass glass = **new** Glass(table, table.getPlacement());  
 Flowers flowers = **new** Flowers(table.getPlacement(), table);  
 Cake cake = **new** Cake(plate.getPlacement(), plate,Human.*getLive*());  
 Clock clock = **new** Clock ();  
 Ligths ligths = **new** Ligths(door,door.getPlacement());  
 Mirror mirror= **new** Mirror();  
 Kolokolchik kolokolchik =**new** Kolokolchik();  
 Kolpachok kolpachok =**new** Kolpachok();  
 Tears tears =**new** Tears();  
 **if** (Philiphonka.*getLive*() == **true**) {  
 Philiphonka.put(table.getPlacement(), table, dish, glass, flowers);  
 System.***out***.println(cake.toString());  
 **if** (Philiphonka.*getIntelligence*() == **true**) {  
 Philiphonka.look(door.getPlacement1(), clock, ligths, ligths.getPlacement(), door, mirror);  
 } **else** {  
 Philiphonka.find();  
 Philiphonka.look(door.getPlacement1(), clock, ligths, ligths.getPlacement(), door, mirror);  
 Philiphonka.find();  
 }  
 Philiphonka.minstay(table.getPlacement(), table);  
 Philiphonka.cry();  
 kolpachok.went(foreHead);  
 kolokolchik.ringing();  
 **while** (Tears.*inDish*() < 7) {  
 System.***out***.print(tears.toString() + tears.getProperty() + tears.getAction());  
 tears.dropTo(dish);  
 }  
 }  
 **else** { System.***out***.println(cake.toString()); }  
 }  
 }

Human:

**package** Laba3;  
**import** Laba3.places.AbstractPlace;  
**import** Laba3.stuff.CommonStuff;  
**public class** Human {  
 **private** String **name**;  
 **private static boolean** *live*;  
 **private static boolean** *intelligence*;  
 **public** Human (String name){  
  
 **this**.**name**=name;  
 }  
 **public void** aLive(){  
 **if** (Math.*random*()<0.05) {  
 System.***out***.print(**"Прости, но "**+ **this**.**name** +**" умрела от депрессии"**);  
 *live* = **false**;}  
 **else**{  
 System.***out***.println(**"Персонаж "**+ **this**.**name** + **" создан "**);  
 *live* = **true**;  
 }  
 }  
 **public static boolean** getLive(){  
 **return** *live*;  
 }  
 **public void** aIntelligence(){  
 **if** (Math.*random*()<0.1){  
 System.***out***.println(**"Прости, но "**+ **this**.**name** +**" глупа,как и создатель этой проги"**);  
 *intelligence* = **false**;}  
 **else** {  
 System.***out***.println(**this**.**name**+**" очень умная"**);  
 *intelligence* = **true**; }  
 }  
 **public static boolean** getIntelligence(){  
 **return** *intelligence*; }  
  
 **public void** put(Placement placement, AbstractPlace place, CommonStuff stuff0,CommonStuff stuff1,CommonStuff stuff2){  
 System.***out***.print(**this**.**name** + **" поставила "** + placement.toString() + place + **" "** + stuff0.toString() +**" , "** + stuff1.toString() + **" и "** +stuff2.toString());  
*//Она поставила на стол тарелки, три стакана и букет цветов, а на плите стоял кекс, совершенно почерневший от ожидания.* }  
 **public void** look(Placement placement0, CommonStuff stuff0,CommonStuff stuff1,Placement placement1, AbstractPlace place,CommonStuff stuff2){  
 System.***out***.print(**this**.**name** + **" взглягула "** + placement0 + stuff0.toString() + **" и "** + stuff1.toString() + placement1 + place + **" , потом посмотрела на себя в "** + stuff2.toString() );  
*//Филифьонка взглянула на часы и гирлянды листьев над дверью, потом посмотрела на себя в зеркало* }  
 **public void** minstay( Placement placement,AbstractPlace place){  
 System.***out***.print(**", оперлась руками "** + placement + place + **" и"** );  
 *//оперлась руками о стол и заплакала.* }  
 **public void** cry(){  
 System.***out***.println(**" заплакала."**);  
  
 }  
 **public void** find(){  
 **if** (Math.*random*()<0.4) {  
 System.***out***.print(**" (она вертится и ищет вещи.)"**);  
 **if** (Math.*random*()<0.4) {  
 System.***out***.println(**" (снова вертится и ищет вещи) "**);  
 }  
 }  
 }  
}

Placement:

**package** Laba3;  
  
**public enum** Placement {  
 ***On***(**" на "**),  
 ***Over***(**" над "**);  
 String **name**;  
 Placement(String name) {  
 **this**.**name** = name;  
 }  
 **public** String toString(){  
 **return name**;  
 }  
}

AbstractPlace:

**package** Laba3.places;  
  
**import** Laba3.Placement;  
  
**public abstract class** AbstractPlace {  
 String **name**;  
  
 **protected** AbstractPlace(String nameplace) {  
 **name**=nameplace;  
 }  
  
 **public abstract** Placement getPlacement();  
  
 **public** String toString(){  
 **return name**;  
 }  
}

Door:

**package** Laba3.places;  
  
**import** Laba3.Placement;  
  
**public class** Door **extends** AbstractPlace{  
  
 **public** Door() {  
 **super**(**"дверь"**);  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return** Placement.***Over***;  
 }  
  
 **public** Placement getPlacement1() {  
 **return** Placement.***On***;  
 }  
}

ForeHead:

**package** Laba3.places;  
  
**import** Laba3.Placement;  
**import** Laba3.stuff.Go;  
  
**public class** ForeHead **extends** AbstractPlace **implements** Go {  
 **public** ForeHead() {  
 **super**(**"лоб"**);  
 }  
  
 @Override  
 **public** AbstractPlace getPlace() {  
 **return null**;  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return** Placement.***On***;  
 }  
  
 @Override  
 **public** String getAction() {  
 **return null**;  
 }  
  
 @Override  
 **public** String getProperty() {  
 **return null**;  
 }  
}

Plate:

**package** Laba3.places;  
  
**import** Laba3.Placement;  
  
**public class** Plate **extends** AbstractPlace {  
  
 **public** Plate() {  
 **super**(**"плита"**);  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return** Placement.***On***;  
 }  
}

Table:

**package** Laba3.places;  
  
**import** Laba3.Placement;  
  
**public class** Table **extends** AbstractPlace {  
 **public** Table() {  
 **super**(**"стол"**);  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return** Placement.***On***;  
 }  
}

Cake:

**package** Laba3.stuff;  
  
**import** Laba3.Placement;  
**import** Laba3.places.AbstractPlace;  
  
**public class** Cake **extends** CommonStuff **implements** StuffWithPlacement,StuffWithProperty {  
 **private** Placement **placement**;  
 **private** AbstractPlace **place**;  
 **private boolean live**;  
  
 **public** Cake(Placement placement, AbstractPlace place, **boolean** live) {  
 **super**(**"кекс"**);  
 **this**.**placement**=placement;  
 **this**.**place**=place;  
 **this**.**live** =live;  
 }  
  
 @Override  
 **public** AbstractPlace getPlace() {  
 **return this**.**place**;  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return this**.**placement**;  
 }  
  
 @Override  
 **public** String getProperty() {  
 **return " совершенно почерневший от ожидания."**;  
 }  
  
 **public** String toString() {  
 **if** (**this**.**live** == **true**) {  
 **return " , а"** + **this**.**placement** + **this**.**place** + **" стоял "** + getName() + **","** + getProperty(); }  
 **else** {  
 **return " , а"** + **this**.**placement** + **this**.**place** + **" стоял "** + getName() + **","** + getProperty() + **" ,в конечном случае разложившись"**; }  
 }  
}

Clock:

**package** Laba3.stuff;  
**public class** Clock **extends** CommonStuff{  
 **public** Clock() {  
 **super**(**"часы"**);  
 }  
 **public int** hashCode (){  
 **return super**.hashCode()\*23-5\*45 ;  
 }  
 **public boolean** equals(Flowers c){  
 **return this**.hashCode() == c.hashCode()? **true**:**false**;  
 }  
}

CommonStuff:

**package** Laba3.stuff;  
**public class** CommonStuff {  
 **private** String **nameStuff**;  
 **public** CommonStuff(String nameStuff) {  
 **this**.**nameStuff** = nameStuff;  
 }  
  
 **public** String getName(){  
 **return this**.**nameStuff**;  
 }  
 **public** String toString(){  
 **return this**.**nameStuff**;  
 }  
}

Dish:

**import** Laba3.places.AbstractPlace;  
  
**public class** Dish **extends** CommonStuff **implements** TearsAcceptor{  
 **private** Placement **placement**;  
 **private** AbstractPlace **place**;  
  
 **public** Dish(Placement placement,AbstractPlace place) {  
 **super**(**"тарелки"**);  
 **this**.**placement**=placement;  
 **this**.**place**=place;  
 }  
 @Override  
 **public** AbstractPlace getPlace() {  
 **return this**.**place**;  
 }  
 @Override  
 **public** Placement getPlacement() {  
 **return this**.**placement**;  
 }  
 @Override  
 **public** String getProperty() {  
 **return "пустую "**;  
 }  
 @Override  
 **public** String getAction() {  
 **return null**;  
 }  
}

Flowers:

**package** Laba3.stuff;  
  
**import** Laba3.Placement;  
**import** Laba3.places.AbstractPlace;  
  
**public class** Flowers **extends** CommonStuff **implements** StuffWithPlacement {  
 **private** Placement **placement**;  
 **private** AbstractPlace **place**;  
  
 **public** Flowers(Placement placement,AbstractPlace place) {  
 **super**(**"цветы"**);  
 **this**.**placement**=placement;  
 **this**.**place**=place;  
 }  
  
 @Override  
 **public** AbstractPlace getPlace() {  
 **return this**.**place**;  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return this**.**placement**;  
 }  
  
 **public int** hashCode (){  
 **return super**.hashCode()\*12-56 ;  
 }  
  
  
}

Glass:

**package** Laba3.stuff;  
  
**import** Laba3.Placement;  
**import** Laba3.places.AbstractPlace;  
  
**public class** Glass **extends** CommonStuff **implements** StuffWithPlacement{  
 **private** AbstractPlace **place**;  
 **private** Placement **placement**;  
  
 **public** Glass(AbstractPlace place,Placement placement) {  
 **super**(**"три стакана"**);  
 **this**.**place**=place;  
 **this**.**placement** =placement;  
 }  
 @Override  
 **public** AbstractPlace getPlace() {  
 **return this**.**place**;  
 }  
 @Override  
 **public** Placement getPlacement() {  
 **return this**.**placement**;  
 }  
}

Go:

**package** Laba3.stuff;  
  
**public interface** Go **extends** StuffWithPlacement,StuffWithAction,StuffWithProperty {  
}

Kolokolchik:

**package** Laba3.stuff;  
  
**public class** Kolokolchik **extends** CommonStuff **implements** StuffWithAction,StuffWithProperty {  
 **private** String **action**;  
  
 **public** Kolokolchik() {  
 **super**(**"колокольчик"**);  
 }  
  
 @Override  
 **public** String getAction() {  
 **return " звякнул "**;  
 }  
  
 @Override  
 **public** String getProperty() {  
 **return "(всего один печальный звук)"**;  
 }  
 **public void** ringing(){  
 System.***out***.print(**this** + **this**.getAction() + **this**.getProperty() + **", и "**);  
 }  
}

Kolpachok:

**package** Laba3.stuff;  
  
**public class** Kolpachok **extends** CommonStuff **implements** StuffWithAction {  
 **private** String **action**;  
  
 **public** Kolpachok() {  
 **super**(**"Колпачок"**);  
 }  
  
 @Override  
 **public** String getAction() {  
 **return " съехал ей "**;  
 }  
  
 **public void** went(Go l){  
 System.***out***.print(**this** + **this**.getAction()+ ((StuffWithPlacement)l).getPlacement()+ l + **", так что "**);  
 }  
}

Ligths:

**package** Laba3.stuff;  
  
**import** Laba3.Placement;  
**import** Laba3.places.AbstractPlace;  
  
**public class** Ligths **extends** CommonStuff **implements** StuffWithPlacement {  
 **private** AbstractPlace **place**;  
 **private** Placement **placement**;  
  
 **public** Ligths(AbstractPlace place,Placement placement) {  
 **super**(**"гирлянды листьев"**);  
 **this**.**place**=place;  
 **this**.**placement** =placement;  
 }  
  
 @Override  
 **public** AbstractPlace getPlace() {  
 **return this**.**place**;  
 }  
  
 @Override  
 **public** Placement getPlacement() {  
 **return this**.**placement**;  
 }  
}

Mirror:

**package** Laba3.stuff;  
  
**public class** Mirror **extends** CommonStuff {  
 **public** Mirror() {  
 **super**(**"зеркало"**);  
 }  
}

StuffWithAction:

**package** Laba3.stuff;  
  
**public interface** StuffWithAction {  
 **public** String getAction();  
}

TearsAcceptor:

**package** Laba3.stuff;  
  
**public interface** TearsAcceptor **extends** StuffWithAction,StuffWithProperty,StuffWithPlacement {  
  
}

StuffWithPlacement:

**package** Laba3.stuff;  
  
**import** Laba3.Placement;  
**import** Laba3.places.AbstractPlace;  
  
**public interface** StuffWithPlacement {  
 **public** AbstractPlace getPlace();  
 **public** Placement getPlacement();  
}

StuffWithProperty:

**package** Laba3.stuff;  
  
**public interface** StuffWithProperty {  
 **public** String getProperty();  
}

Tears:

**package** Laba3.stuff;  
  
**public class** Tears **extends** CommonStuff **implements** StuffWithAction,StuffWithProperty {  
 **private** String **action**;  
 **private static int** *i*=1 ;  
  
 **public** Tears() {  
 **super**(**"слезы"**);  
 }  
  
 @Override  
 **public** String getAction() {  
 **return "закапали"**;  
 }  
  
 @Override  
 **public** String getProperty() {  
 **return " медленно "**;  
 }  
  
 **public static int** inDish(){  
 *i* = (**int**) (*i* + Math.*random*()\*5);  
 **return** *i*;  
 }  
 **public void** dropTo(TearsAcceptor t){  
 System.***out***.print( t.getPlacement() + t.getProperty() + t.toString()+**" . "**);  
 }  
}

Вывод программы:

Персонаж Филифьонка создан

Филифьонка поставила на стол тарелки , три стакана и цветы , а на плита стоял кекс, совершенно почерневший от ожидания.

Филифьонка взглягула на часы и гирлянды листьев над дверь , потом посмотрела на себя в зеркало, оперлась руками на стол и заплакала.

Колпачок съехал ей на лоб, так что колокольчик звякнул (всего один печальный звук), и слезы медленно закапали на пустую тарелки .

Вывод:

Я научился работать с интерфейсами почитал про такие функции в java, как enum, equals(), toString() и hashCode(). Теперь я лучше разбираюсь в ООП и могу делать объектные модели.