Национальный исследовательский университет ИТМО

Факультет СУиР

Лабораторная работа № 3

Работу выполнил:

Малышева Анна

Группа № R3137

Преподаватель:

Райла Мартин

г. Санкт-Петербург

2020

**Вариант: 373052**

**Задание:**

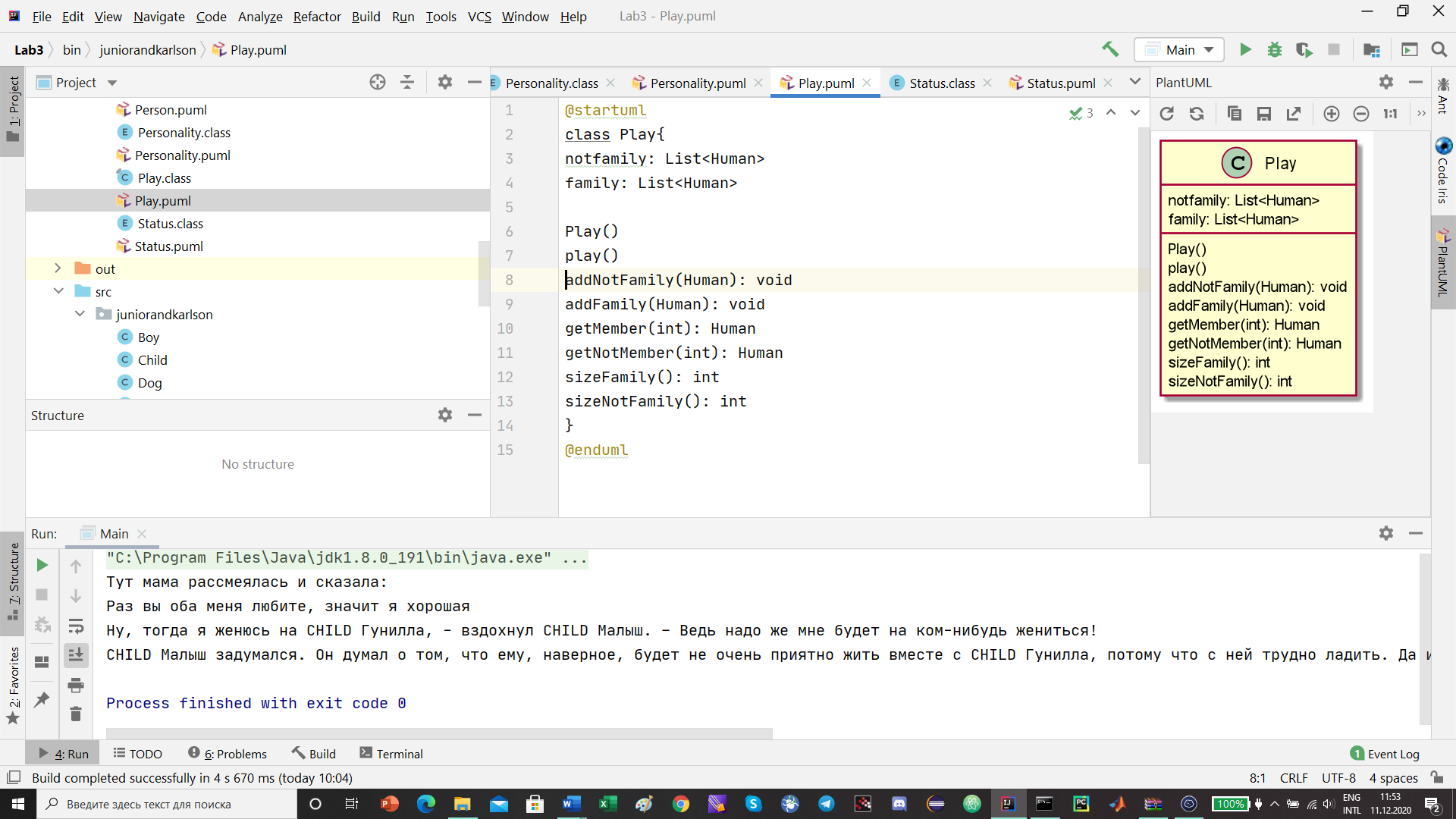
**Описание предметной области, по которой должна быть построена объектная модель:**

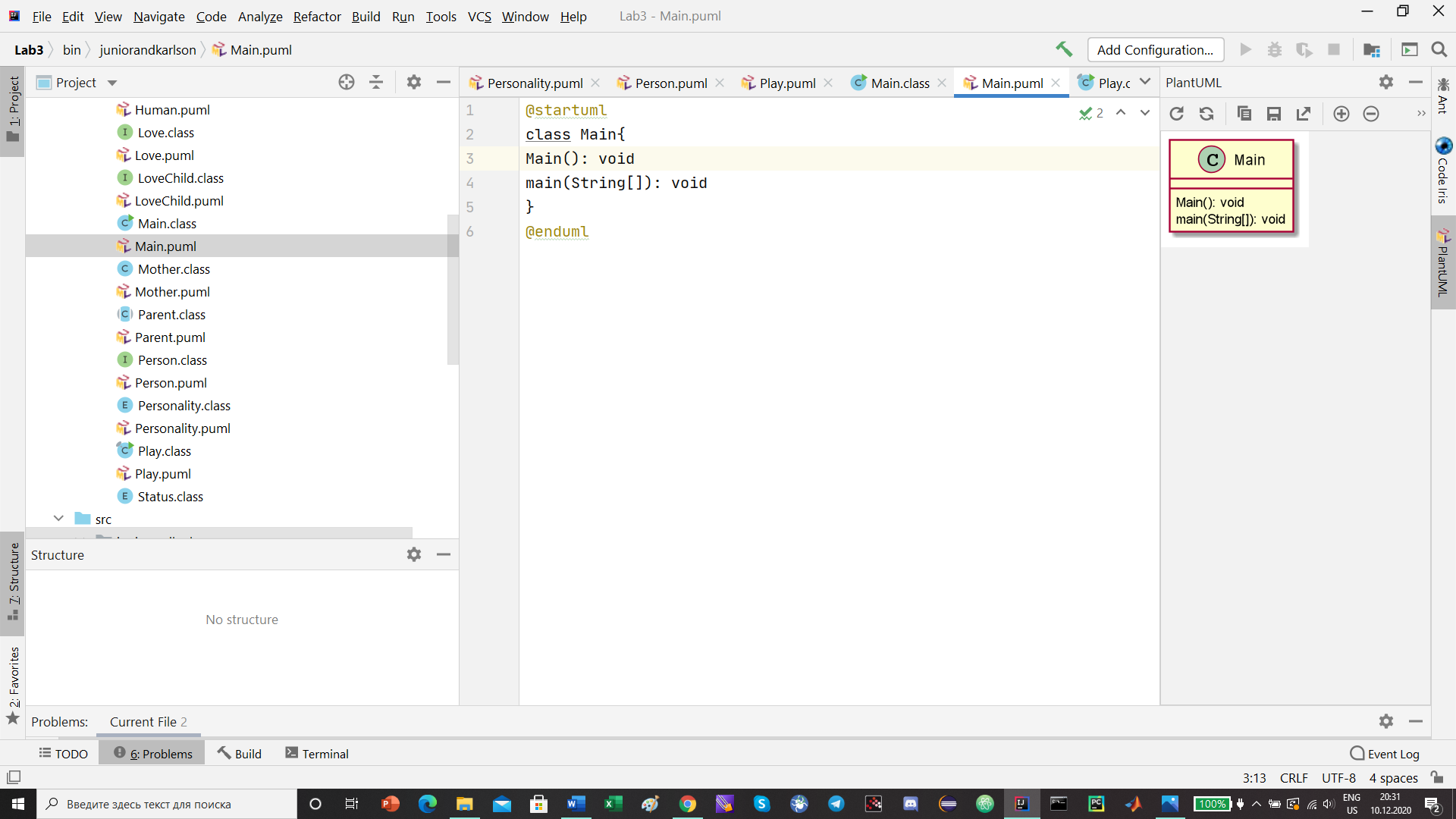
Тут мама рассмеялась и сказала:

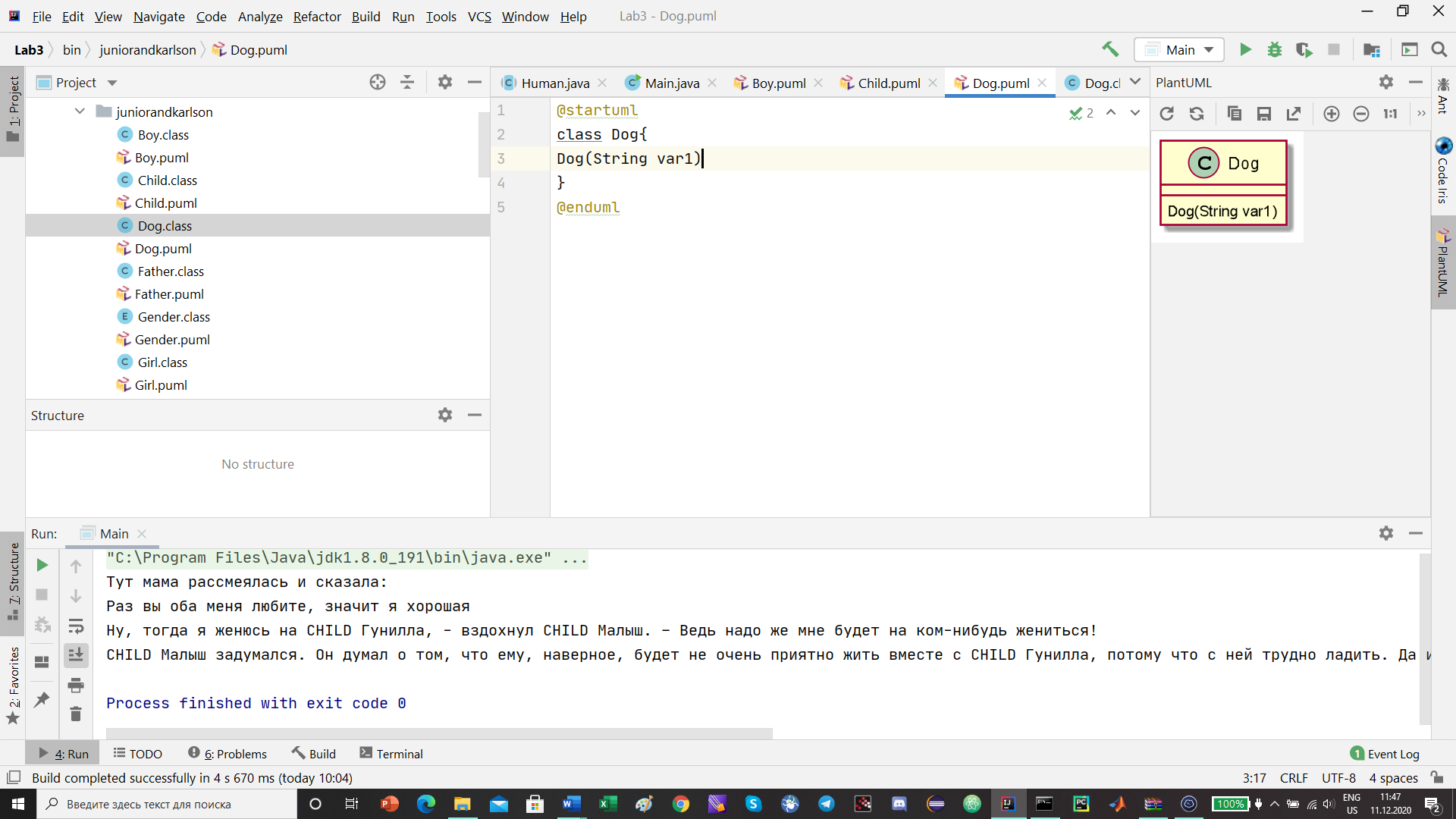
– Раз вы оба меня любите, значит, я хорошая.

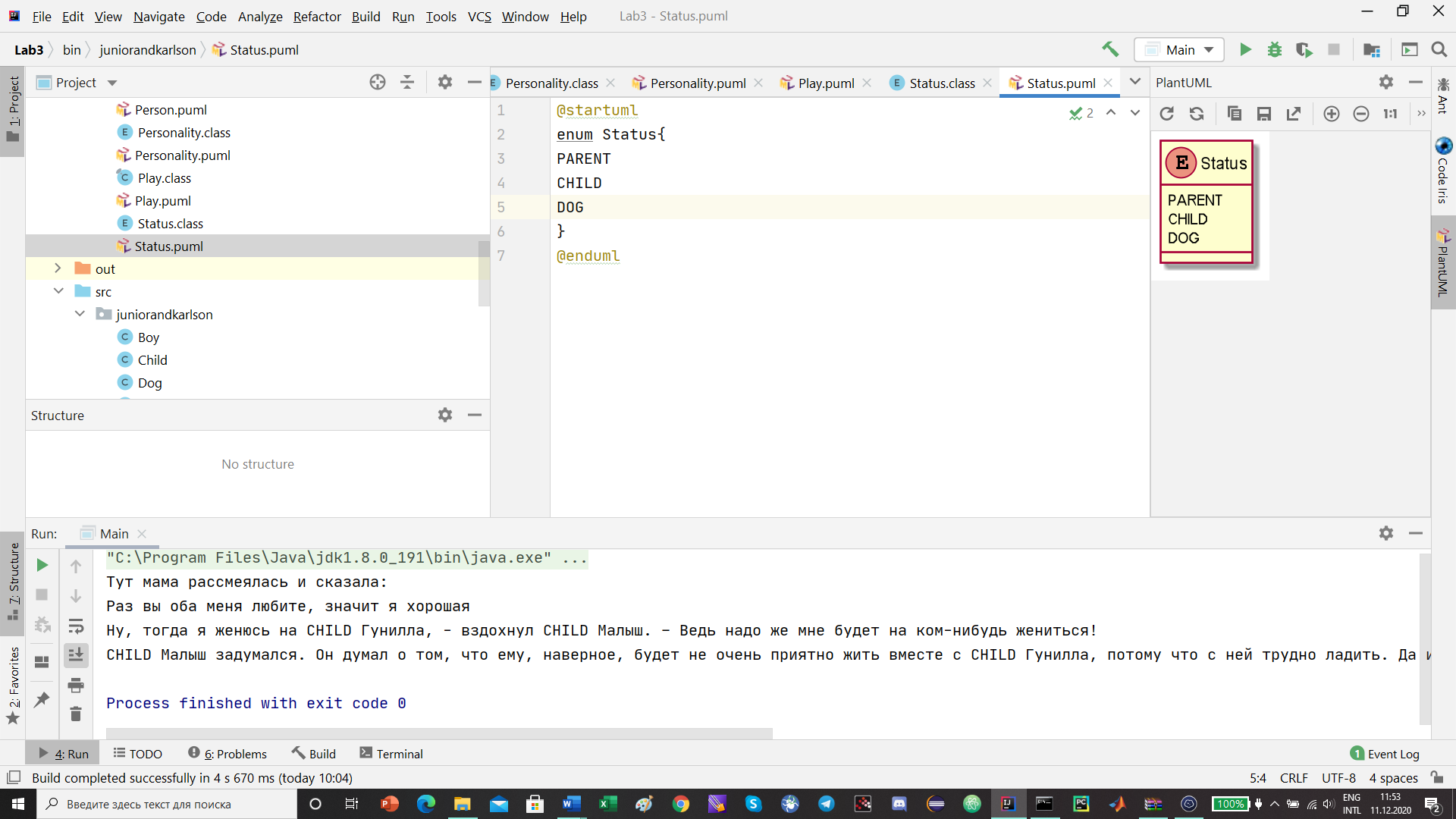
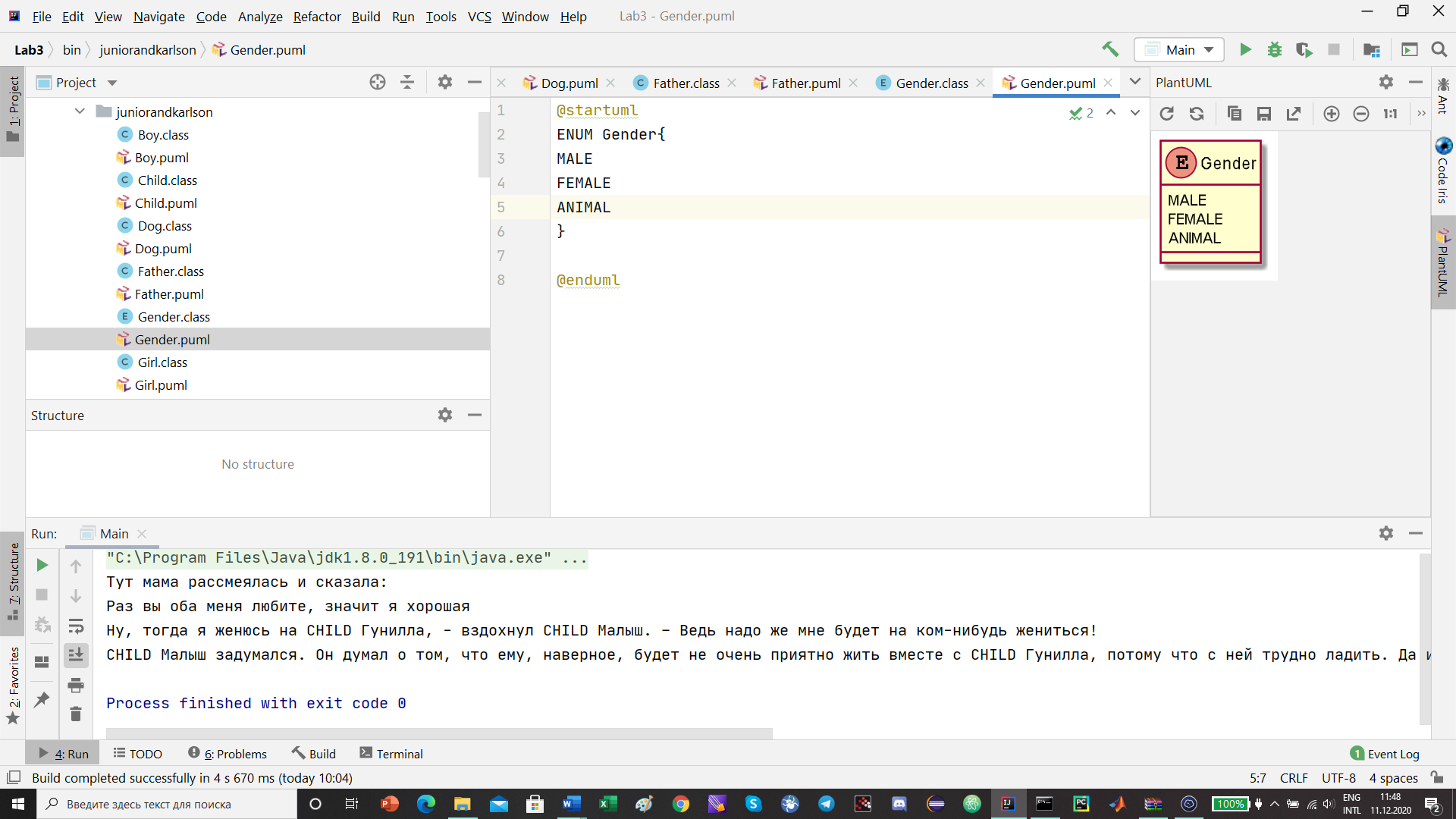
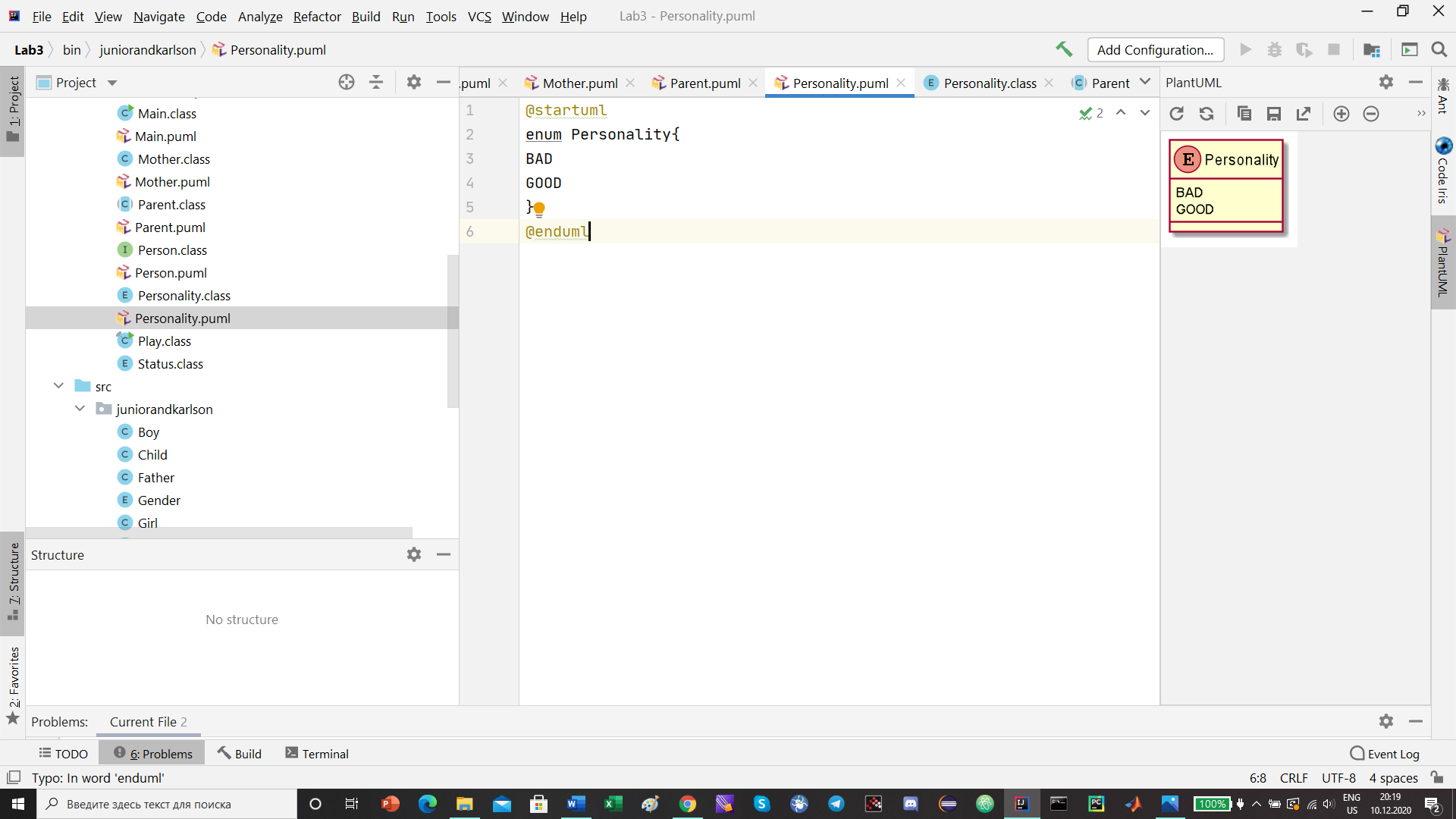
– Ну, тогда я женюсь на Гунилле, – вздохнул Малыш. – Ведь надо же мне будет на ком-нибудь жениться!

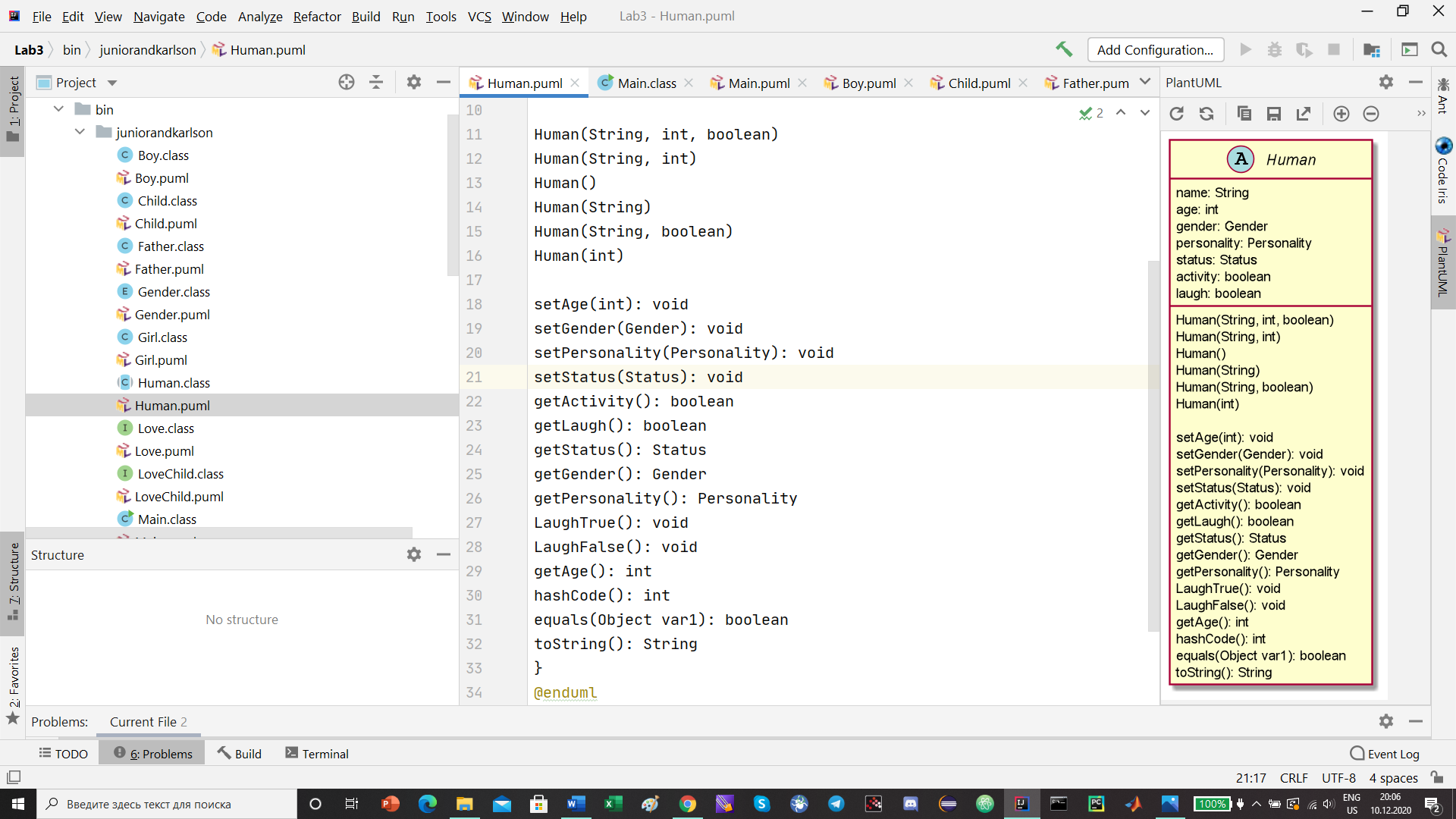
И Малыш вновь задумался. Он думал о том, что ему, наверно, будет не очень приятно жить вместе с Гуниллой, потому что с ней иногда трудно ладить. Да и вообще ему больше всего хотелось жить вместе с мамой, папой, Боссе и Бетан, а не с какой-то там женой.

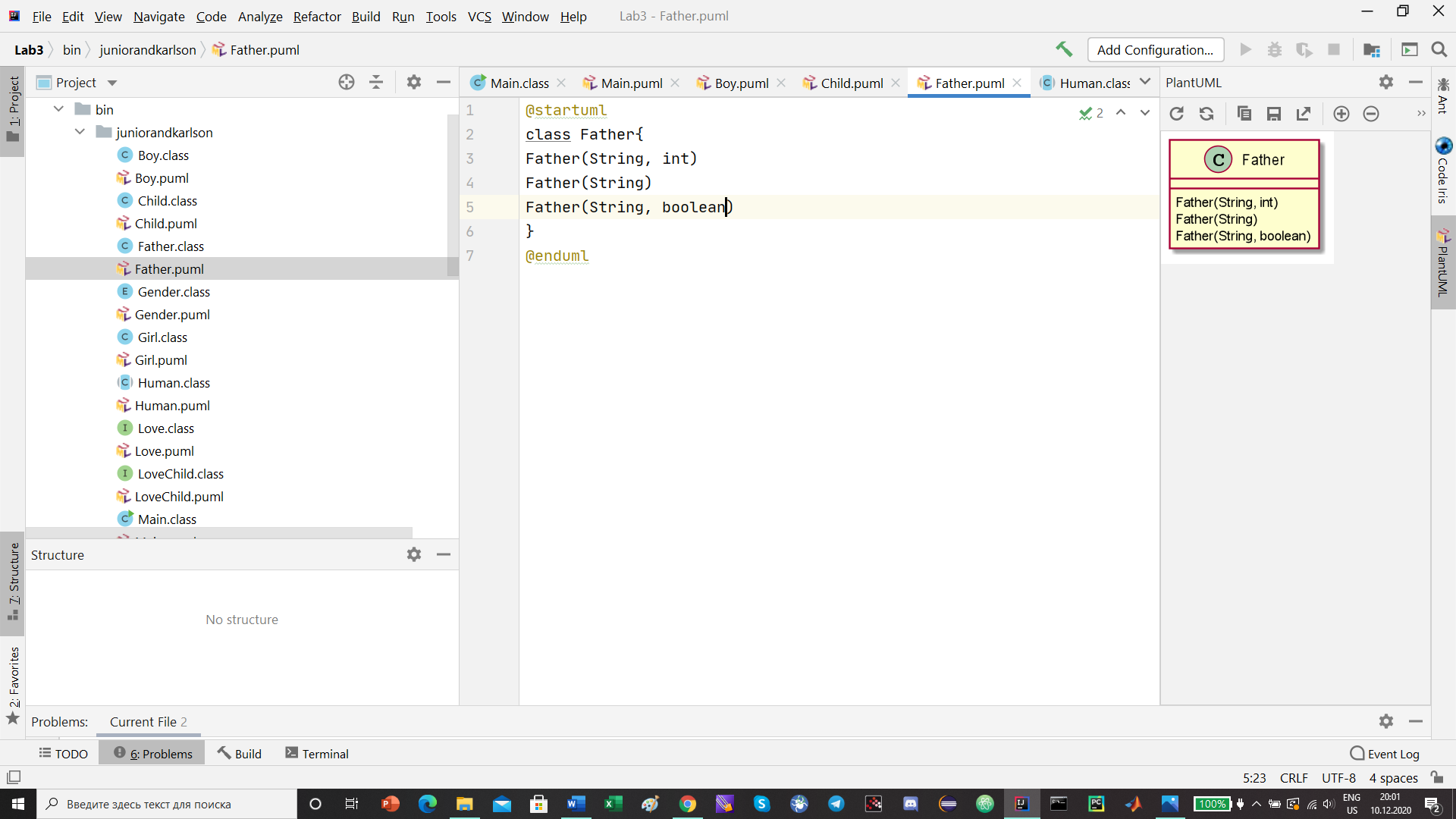
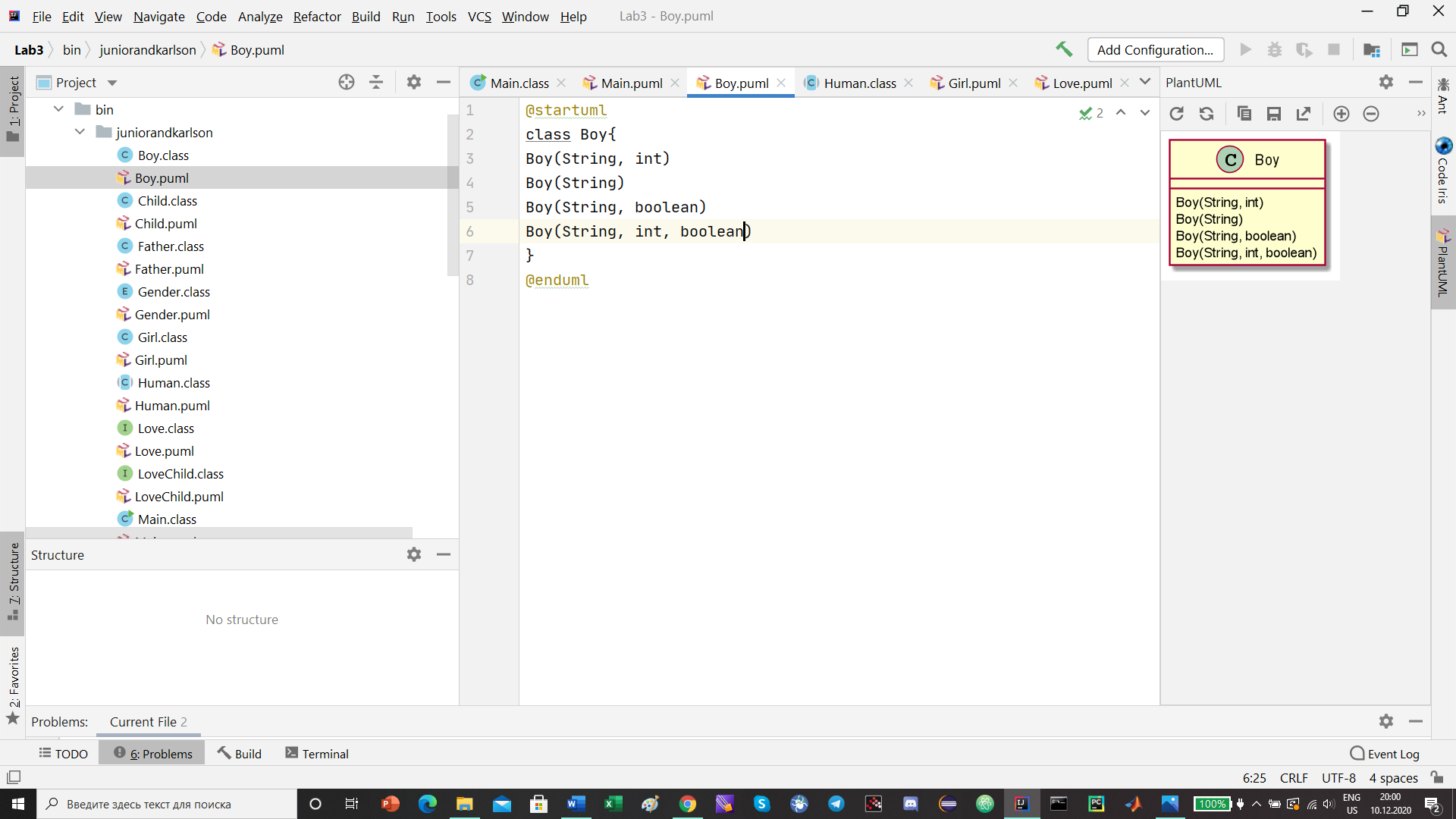
**Диаграмма классов**

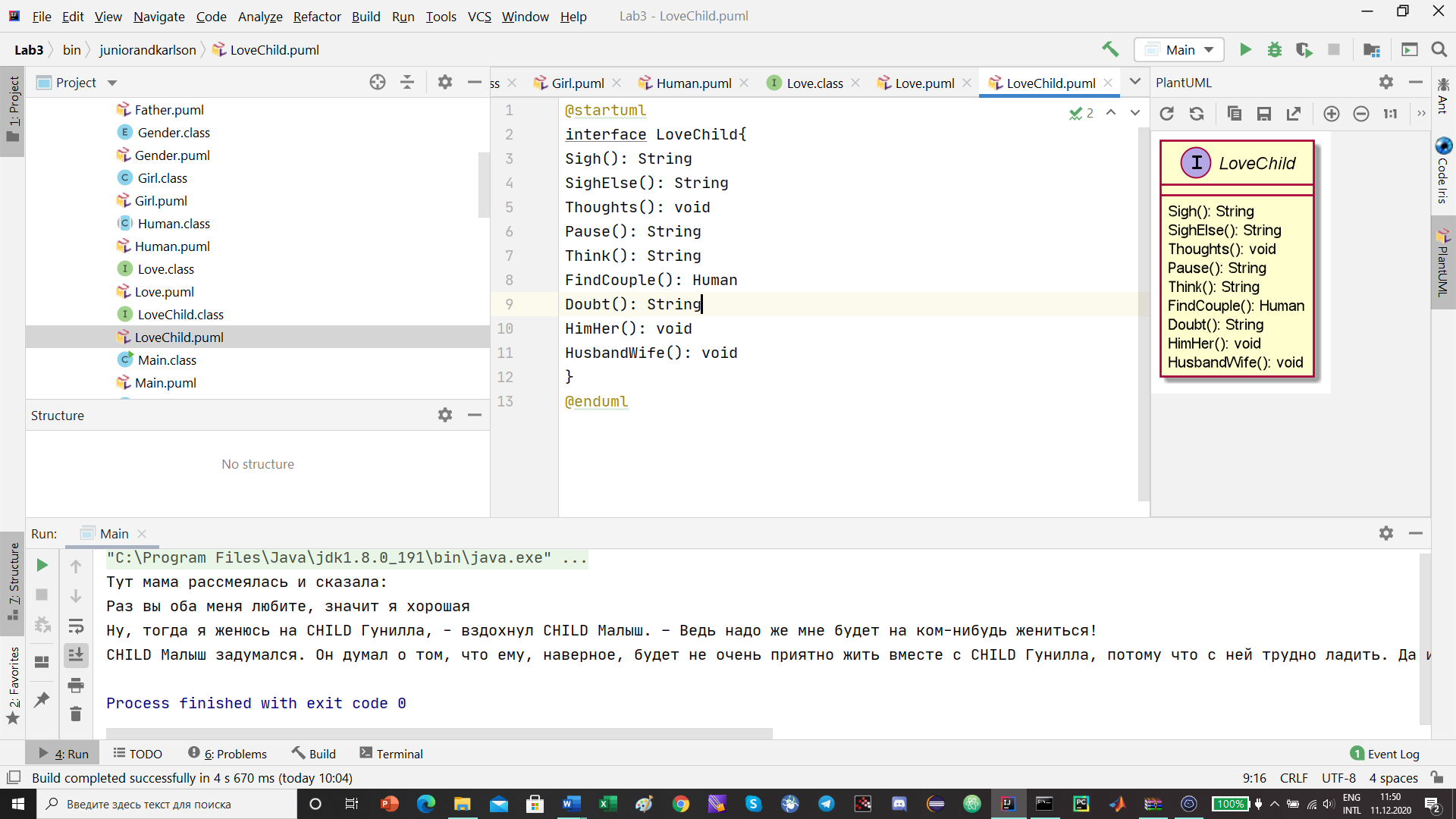


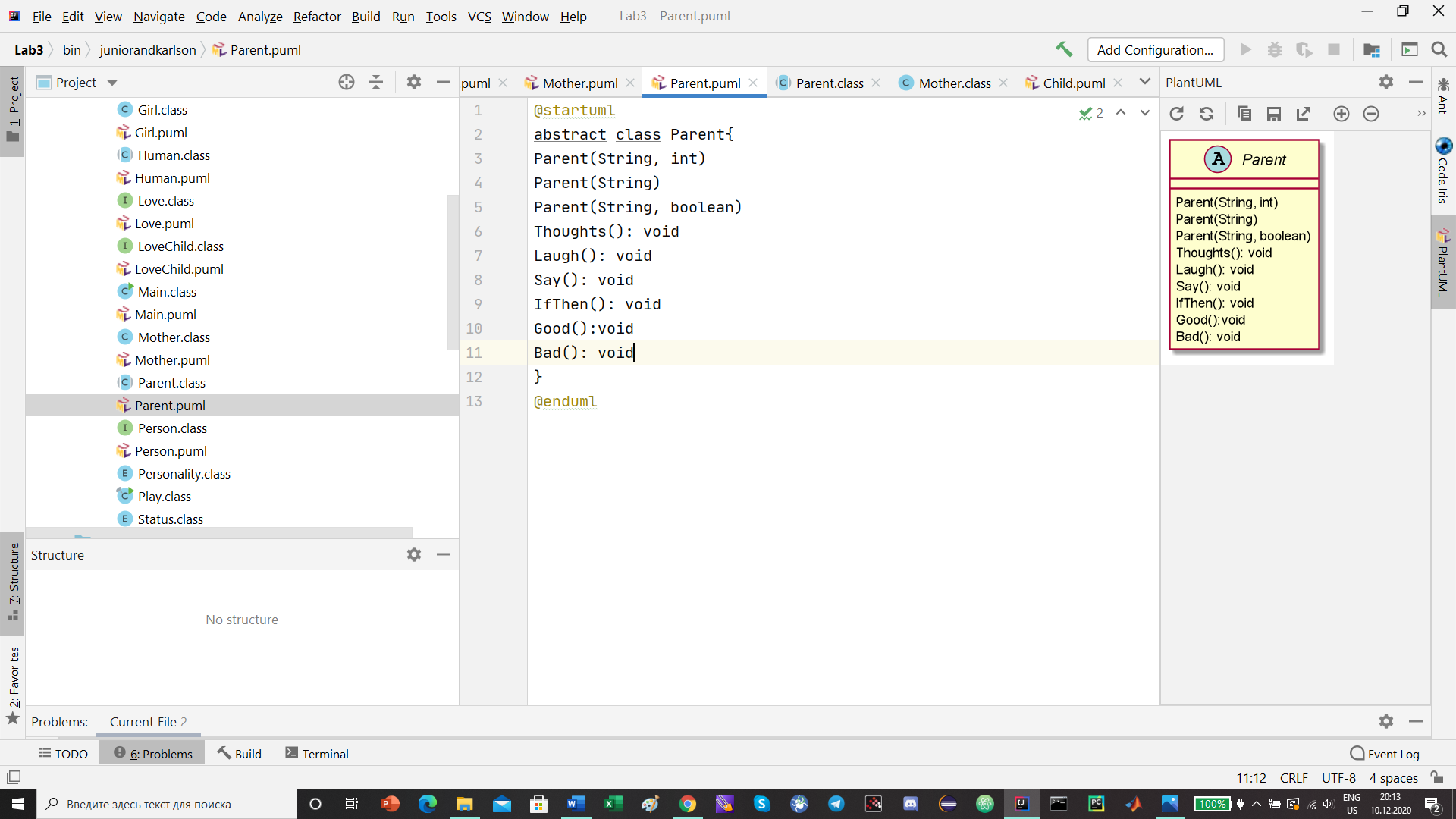


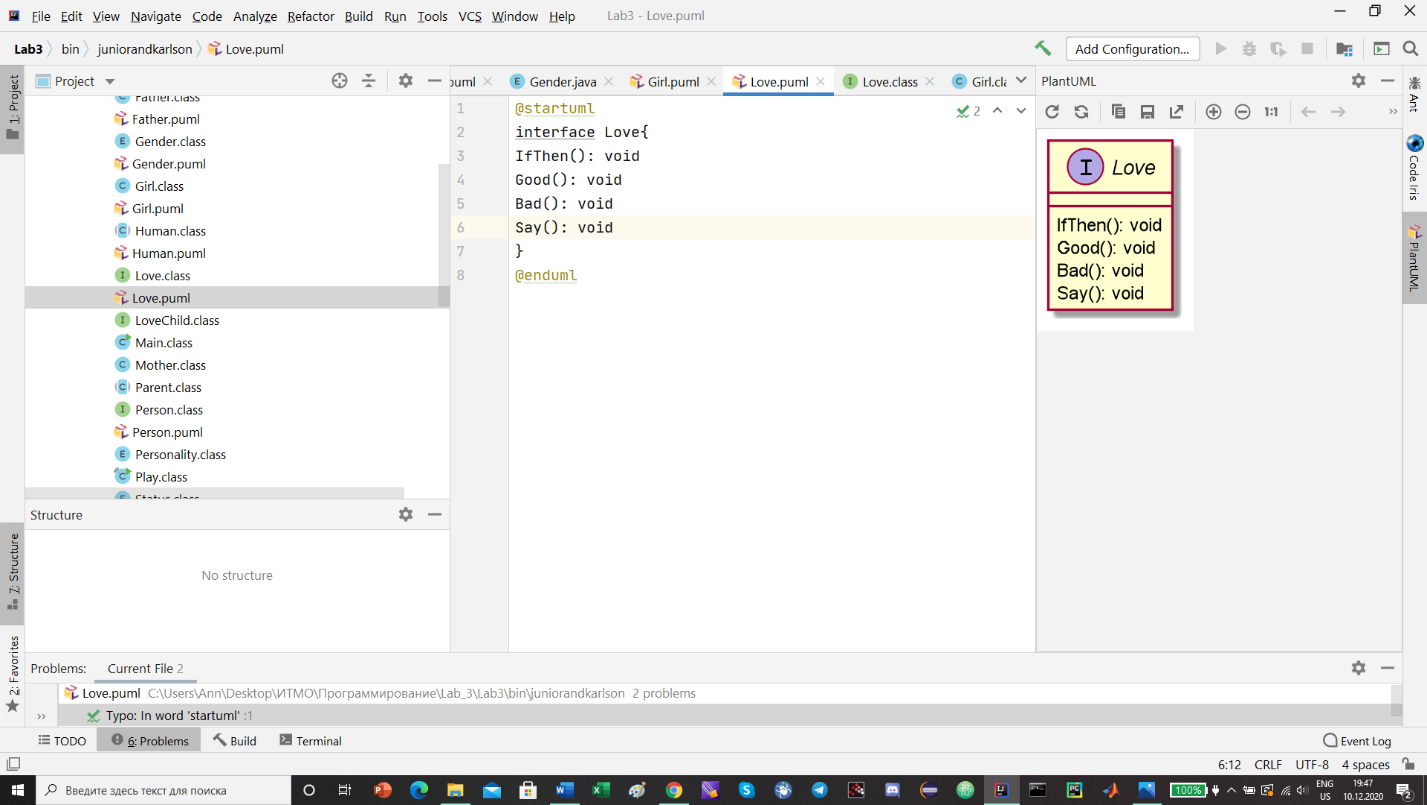
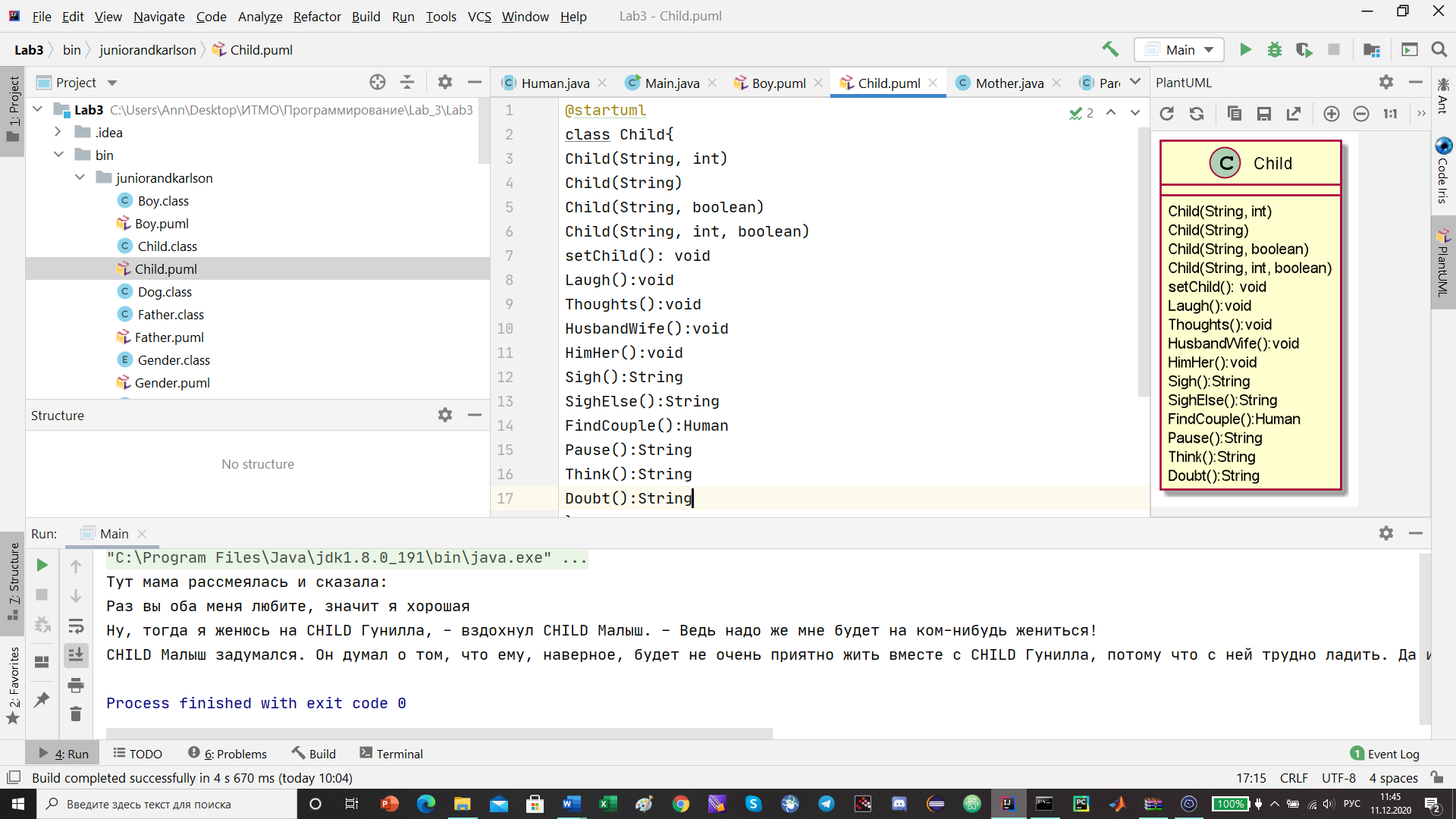


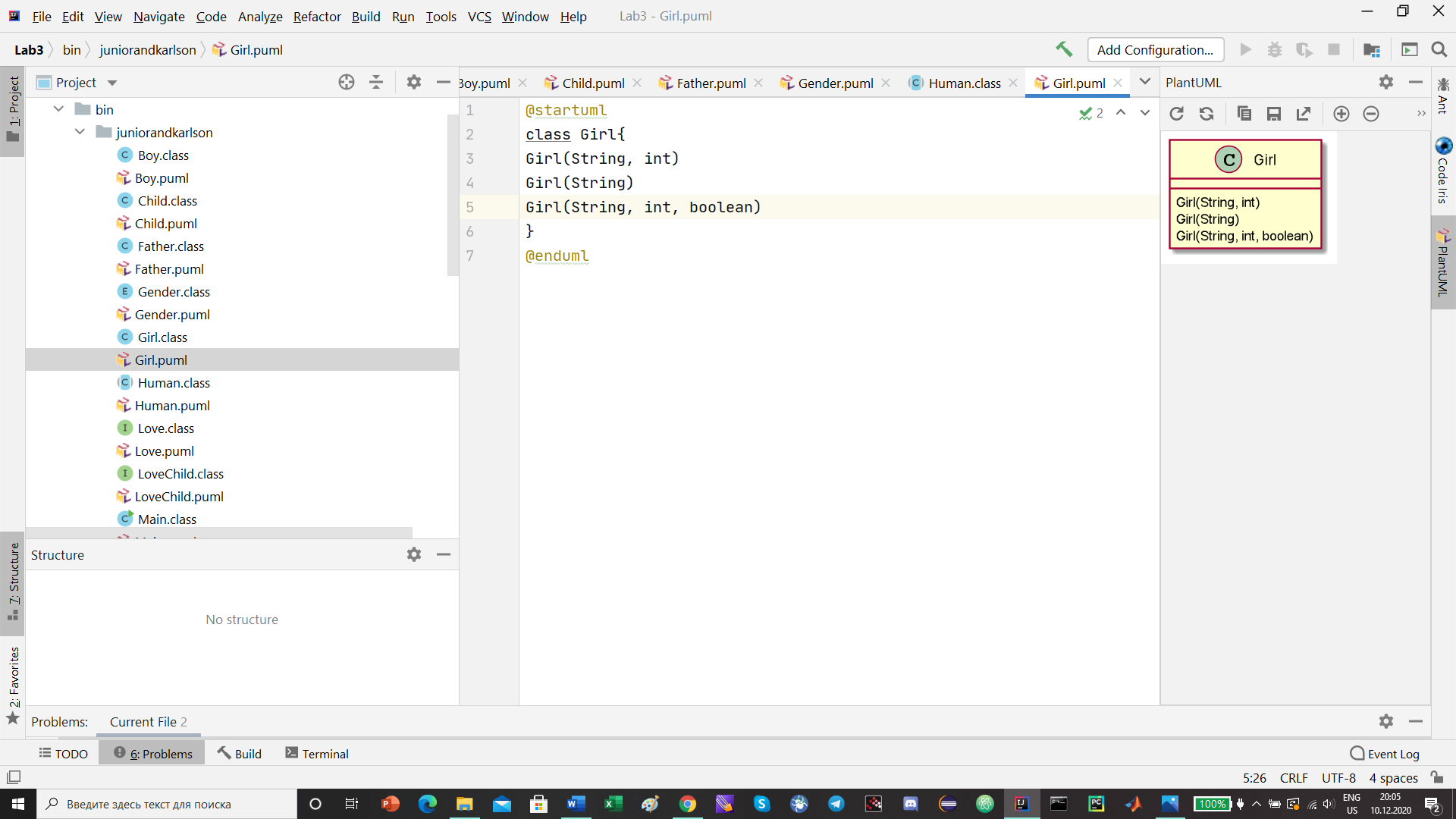


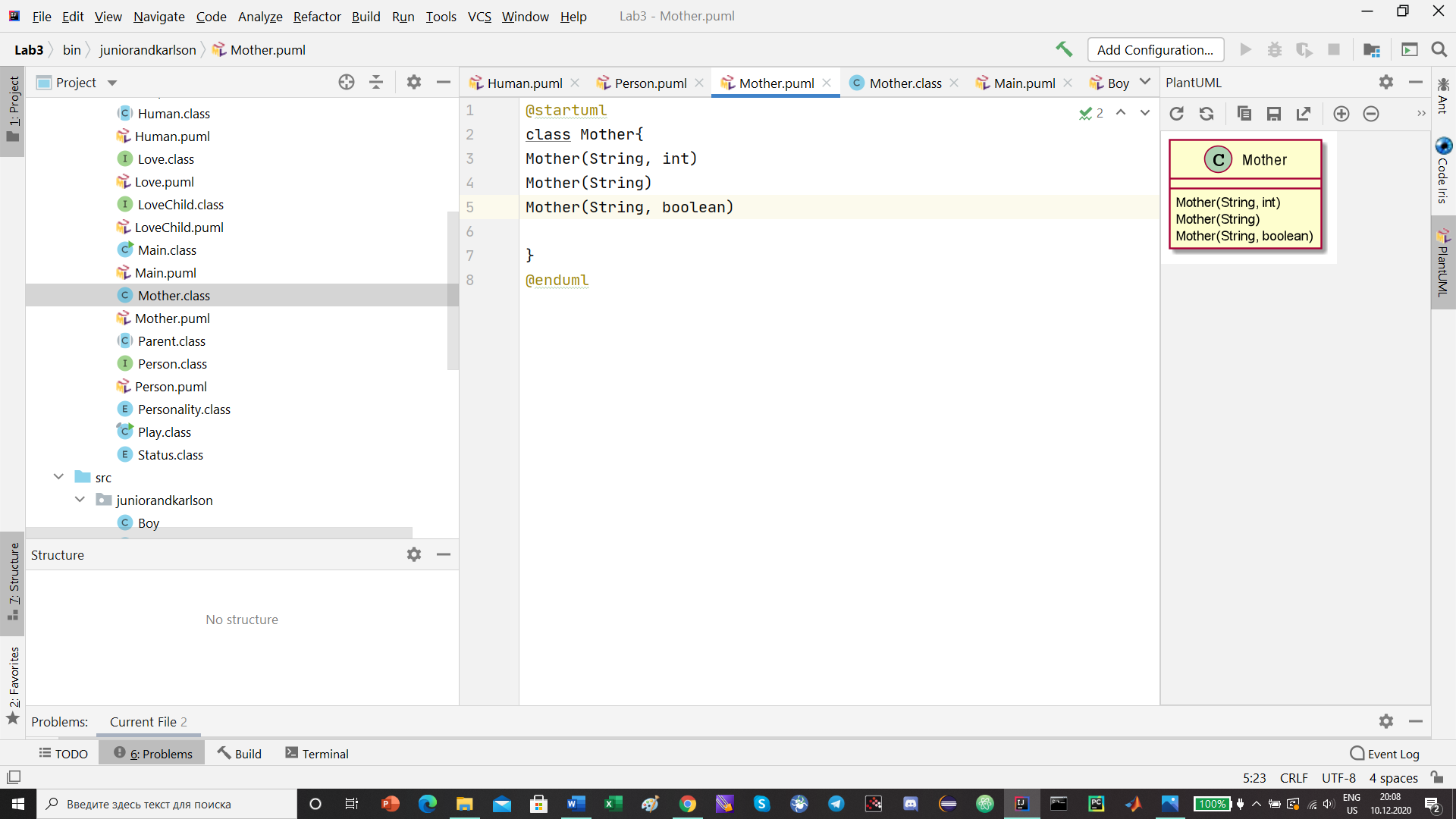


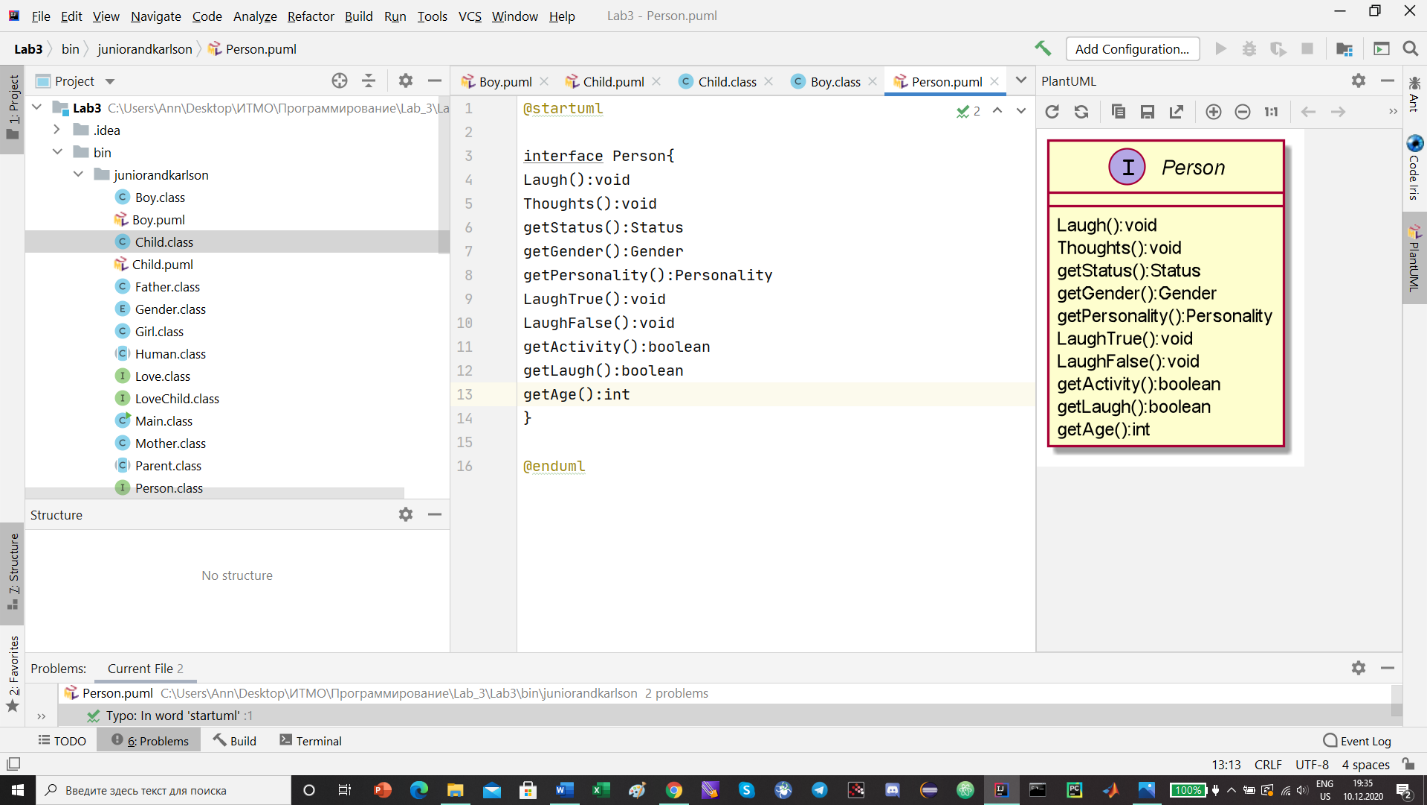












**Исходный код программы**

1. Boy

package juniorandkarlson;  
  
public class Boy extends Child{  
 public Boy(String name, int age) {  
 super(name, age);  
 setGender(Gender.*MALE*);  
 }  
  
 public Boy(String name) {  
 super(name);  
 setGender(Gender.*MALE*);  
 }  
 public Boy(String name, boolean b) {  
 super(name, b);  
 setGender(Gender.*MALE*);  
 }  
  
 public Boy(String name, int age, boolean b) {  
 super(name, age, b);  
 setGender(Gender.*MALE*);  
 }  
  
}

1. Child

package juniorandkarlson;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Child extends Human implements LoveChild{  
 public Child(String name, int age){  
 super(name, age);  
 this.setChild();  
 }  
 public Child(String name){  
 super(name);  
 this.setChild();  
 }  
 public Child(String name, boolean b) {  
 super(name, b);  
 this.setChild();  
 }  
 public Child(String name, int age, boolean b) {  
 super(name, age, b);  
 this.setChild();  
 }  
  
 public void setChild(){  
 this.setStatus(Status.*CHILD*);  
 this.setGender(Gender.*MALE*);  
 if (this.getAge() < 13) {  
 this.setPersonality(Personality.*GOOD*);  
 }  
 }  
  
 @Override  
 public void Laugh() {  
 }  
  
 @Override  
 public void Thoughts() {  
 Human h = FindCouple();  
 if (h == null){  
 System.*out*.println(Sigh());  
 }else {  
 System.*out*.print(SighElse() + Pause() + Think() + Doubt());

System.*out*.print(" будет не очень приятно жить вместе с " + h.toString());  
 if (h.getPersonality() == Personality.*BAD*) {  
 if (h.getGender() == Gender.*FEMALE*) {  
 System.*out*.print(", потому что с ней трудно ладить. ");  
 } else {  
 System.*out*.print(", потому что с ним трудно ладить. ");  
 }  
 } else {  
 if (h.getGender() == Gender.*FEMALE*) {  
 System.*out*.print(", хотя с ней и дружу. ");  
 } else {  
 System.*out*.print(", хотя с ним и дружу. ");  
 }  
 }  
 System.*out*.print("Да и вообще е");  
 }  
 HimHer();  
 System.*out*.print("больше всего хотелось жить вмсете с ");  
 List l = new ArrayList();  
 for (int i = 0; i < Play.*sizeFamily*(); i++){  
 if (!Play.*getMember*(i).equals(this)){  
 l.add(Play.*getMember*(i));  
 }  
 }  
 if (l.size() > 1) {  
 for (int i = 0; i < l.size()-2; i++) {  
 System.*out*.print(l.get(i).toString());  
 System.*out*.print(", ");  
 }  
 System.*out*.print(l.get(l.size()-2).toString() + " и " + l.get(l.size()-1).toString());  
 }else{  
 System.*out*.println(l.get(0).toString());  
 }  
 HusbandWife();  
  
 }  
  
 public void HusbandWife() {  
 System.*out*.println(", а не с какой-то там женой.");  
 }  
  
 public void HimHer() {  
 System.*out*.print("му ");  
 }  
  
 @Override  
 public String Sigh() {  
 return "Ну, тогда я ни на ком не женюсь, - вздохнул " + this.toString() + ".\nЕ";  
 }  
  
 @Override  
 public String SighElse(){  
 return "Ну, тогда я женюсь на " + FindCouple().toString() + ", - вздохнул " + this.toString() + ". – Ведь надо же мне будет на ком-нибудь жениться!\n";  
 }  
  
 @Override  
 public Human FindCouple(){  
 Human h = null;  
 for (int i = 0; i < Play.*sizeNotFamily*(); i++){  
 if (Play.*getNotMember*(i).getGender() == Gender.*FEMALE*){  
 h = Play.*getNotMember*(i);  
 }  
 break;  
 }  
 return h;  
 }  
  
 @Override  
 public String Pause(){  
 return this.toString() + " задумался. ";  
 }  
  
 @Override  
 public String Think(){  
 return "Он думал о том, что ему";  
 }  
  
 @Override  
 public String Doubt(){  
 return ", наверное,";

}  
  
}

1. Father

package juniorandkarlson;  
  
public class Father extends Parent {  
 public Father(String name, int age) {  
 super(name, age);  
 setGender(Gender.*MALE*);  
 }  
  
 public Father(String name) {  
 super(name);  
 setGender(Gender.*MALE*);  
 }  
  
 public Father(String name, boolean b) {  
 super(name, b);  
 setGender(Gender.*MALE*);  
 }  
  
  
}

1. Gender

package juniorandkarlson;  
  
public enum Gender {  
 *MALE*,  
 *FEMALE*,

*ANIMALE*;  
  
 private Gender(){  
 }  
  
}

1. Girl

package juniorandkarlson;  
  
public class Girl extends Child{  
 public Girl(String name, int age) {  
 super(name, age);  
 setGender(Gender.*FEMALE*);  
 }  
  
 public Girl(String name) {  
 super(name);  
 setGender(Gender.*FEMALE*);  
 }  
  
 public Girl(String name, int i, boolean b) {  
 super(name, i, b);  
 this.setGender(Gender.*FEMALE*);  
 }  
  
 @Override  
 public String Sigh() {  
 return "Ну, тогда я ни за кого не выйду замуж, - вздохнула " + this.toString() + ".\nЕ";  
 }  
  
 @Override  
 public String SighElse(){  
 return "Ну, тогда я выйду замуж за " + FindCouple().toString() + ", - вздохнула " + this.toString() + ". – Ведь надо же мне будет выйти замуж за кого-нибудь!\n";  
 }  
  
 @Override  
 public String Pause(){  
 return "И " + this.toString() + "задумалась. ";  
 }  
  
 @Override  
 public String Think(){  
 return "Она думала о том, что ей";  
 }  
  
 @Override  
 public Human FindCouple(){  
 Human h = null;  
 for (int i = 0; i < Play.*sizeNotFamily*(); i++){  
 if (Play.*getNotMember*(i).getGender() == Gender.*MALE*){  
 h = Play.*getNotMember*(i);  
 }  
 break;  
 }  
 return h;  
 }  
 public void HimHer(){  
 System.*out*.print("й ");  
 }  
 public void HusbandWife(){  
 System.*out*.print(", а не с каким-то там мужем.");  
 }  
  
}

1. Human

package juniorandkarlson;  
  
public abstract class Human implements Person{  
  
 private String name = "noname";  
 private int age = 20;  
 private Gender gender = Gender.*MALE*;  
 private Personality personality = Personality.*BAD*;  
 private Status status = Status.*CHILD*;  
 private boolean activity = false;  
 private boolean laugh = false;  
  
 public Human (String var1, int var2, boolean a){  
 this.name = var1;  
 setAge(var2);  
 this.activity = a;  
 }  
  
 public Human (String var1, int var2){  
 this.name = var1;  
 this.setAge(var2);  
 }  
  
 public Human(){  
 }  
  
 public Human(String var1){  
 this.name = var1;  
 }  
  
 public Human(String var1, boolean a){  
 this.name = var1;  
 this.activity = a;  
 }  
  
 public Human(int var1){  
 this.setAge(var1);  
 }  
  
 public void setAge(int var1){  
 if (var1 < 0){  
 var1 = 20;  
 }  
  
 if (var1 > 99){  
 var1 = 99;  
 }  
 this.age = var1;  
 }  
  
 public void setGender(Gender var1){  
 if (var1 == null){  
 var1 = Gender.*MALE*;  
 }  
 this.gender = var1;  
 }  
  
 public void setPersonality(Personality var1){  
 if (var1 == null){  
 var1 = Personality.*BAD*;  
 }  
 this.personality = var1;  
 }  
  
 public void setStatus(Status var1){  
 if (var1 == null){  
 var1 = Status.*CHILD*;  
 }  
 this.status = var1;  
 }  
  
 @Override  
 public boolean getActivity(){  
 return(this.activity);  
 }  
  
 @Override  
 public boolean getLaugh() {  
 return this.laugh;  
 }  
  
 @Override  
 public Status getStatus(){ return this.status; }  
  
 @Override  
 public Gender getGender(){ return this.gender;}  
  
 @Override  
 public Personality getPersonality(){  
 return this.personality;  
 }  
  
 @Override  
 public void LaughTrue(){  
 this.laugh = true;  
 }  
  
 @Override  
 public void LaughFalse(){  
 this.laugh = false;  
 }  
  
 @Override  
 public int getAge(){ return this.age; }  
  
 @Override  
 public int hashCode(){  
 int a = this.age;  
 if (this.gender == Gender.*FEMALE*){  
 a += 100;  
 }  
 if (this.gender == Gender.*MALE*){  
 a += 200;  
 }  
 if (this.status == Status.*CHILD*){  
 a += 1000;  
 }  
 if (this.status == Status.*PARENT*){  
 a += 2000;  
 }  
 return a;  
 }  
  
 @Override  
 public boolean equals(Object h){  
 return (h.hashCode() == this.hashCode());  
 }  
  
 @Override  
 public String toString(){  
 return (this.status + " " + this.name);  
 }  
  
}

1. Love

package juniorandkarlson;  
  
public interface Love {  
 void IfThen();  
 void Good();  
 void Bad();  
 void Say();  
  
  
}

1. LoveChild

package juniorandkarlson;  
  
public interface LoveChild {  
 String Sigh();  
 String SighElse();  
 String Pause();  
 void Think();  
 Human FindCouple();  
 String Doubt();  
 void HimHer();  
 void HusbandWife();  
}

1. Main

package juniorandkarlson;  
  
public class Main {  
  
 public static void main(String[] args){  
 Play p = new Play();  
 Mother m = new Mother("Мама", true);  
 Father f = new Father("Папа");  
 Boy j = new Boy("Малыш", 8, true);  
 Boy b = new Boy("Боссе", 15);  
 Girl g = new Girl("Бетан", 14);  
 Girl gu = new Girl("Гунилла");  
  
 p.addFamily(m);  
 p.addFamily(f);  
 p.addFamily(j);  
 p.addFamily(b);  
 p.addFamily(g);  
 p.addNotFamily(gu);  
  
  
 p.play();  
 }  
}

1. Mother

package juniorandkarlson;  
  
public class Mother extends Parent{  
  
 public Mother(String name, int age) {  
 super(name, age);  
 setGender(Gender.*FEMALE*);  
 }  
  
 public Mother(String name) {  
 super(name);  
 setGender(Gender.*FEMALE*);  
 }  
  
 public Mother(String name, boolean b) {  
 super(name, b);  
 setGender(Gender.*FEMALE*);  
 }  
  
}

1. Parent

package juniorandkarlson;  
  
  
public abstract class Parent extends Human implements Love, Person{  
  
 public Parent(String name, int age){  
 super(name, age);  
 this.setStatus(Status.*PARENT*);  
 }  
 public Parent(String name){  
 super(name);  
 this.setStatus(Status.*PARENT*);  
 }  
  
 public Parent(String name, boolean b) {  
 super(name, b);  
 this.setStatus(Status.*PARENT*);  
 this.setPersonality(Personality.*GOOD*);  
 }  
  
 @Override  
 public void Thoughts() {  
 }  
  
  
 @Override  
 public void Laugh() {  
 this.LaughTrue();  
 if (this.getGender() == Gender.*FEMALE*) {  
 System.*out*.print("Тут мама рассмеялась ");  
 }  
 if (this.getGender() == Gender.*MALE*) {  
 System.*out*.print("Тут папа рассмеялся ");  
 }  
 Say();  
 IfThen();  
 }  
  
  
 @Override  
 public void Say() {  
 if (this.getLaugh()){  
 if (this.getGender() == Gender.*FEMALE*) {  
 System.*out*.println("и сказала:");  
 }  
 if (this.getGender() == Gender.*MALE*) {  
 System.*out*.println("и сказал:");  
 }  
 LaughFalse();  
 }  
 }  
  
 @Override  
 public void IfThen() {  
 if (this.getActivity()) {  
 System.*out*.print("Раз ");  
 int n = 0;  
 Human h = null;  
 for (int i = 0; i < Play.*sizeFamily*(); i++) {  
 if ((Play.*getMember*(i).hashCode() > 2200) || ((Play.*getMember*(i).getStatus() == Status.*CHILD*) && (Play.*getMember*(i).getPersonality() == Personality.*GOOD*))) {  
 n += 1;  
 h = Play.*getMember*(i);  
 }  
 }  
 switch (n) {  
 case 1:  
 if (h.getActivity()) {  
 System.*out*.print("ты меня любишь, значит я ");  
 } else {  
 System.*out*.print("кто-то один меня любит, значит я ");  
 }  
 Good();  
 break;  
 case 2:  
 System.*out*.print("вы оба меня любите, значит я ");  
 Good();  
 break;  
 case 0:  
 System.*out*.print("никто меня не любит, значит я ");  
 Bad();  
 break;  
 default:  
 System.*out*.print("много людей меня любят, значит я ");  
 Good();  
 break;  
 }  
 System.*out*.println();  
 }  
 }  
  
  
 @Override  
 public void Good(){  
 if (this.getGender() == Gender.*FEMALE*) {  
 System.*out*.print("хорошая");  
 } else {  
 System.*out*.print("хороший");  
 }  
 }  
 public void Bad(){  
 if (this.getGender() == Gender.*FEMALE*) {  
 System.*out*.print("плохая");  
 } else {  
 System.*out*.print("плохой");  
 }  
 }  
  
}

1. Person

package juniorandkarlson;  
  
public interface Person {  
 void Laugh();  
 void Thoughts();  
 Status getStatus();  
 Gender getGender();  
 Personality getPersonality();  
 void LaughTrue();  
 void LaughFalse();  
 boolean getActivity();  
 boolean getLaugh();  
 int getAge();  
}

1. Personality

package juniorandkarlson;  
  
public enum Personality {  
 *GOOD*,  
 *BAD*;  
  
 private Personality(){  
 }  
  
}

1. Play

package juniorandkarlson;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public final class Play {  
  
 private static List<Human> *notfamily*;  
 private static List<Human> *family*;  
  
 public Play(){  
 this.*notfamily* = new ArrayList();  
 this.*family* = new ArrayList();  
 }  
  
 public void addNotFamily(Human var1){  
 this.*notfamily*.add(var1);  
 }  
  
 public void addFamily(Human var1){  
 this.*family*.add(var1);  
 }  
  
 public void play(){  
 for (int i = 0; i < this.*family*.size(); i++){  
 if ((*family*.get(i).hashCode() > 2000) && (*family*.get(i).getActivity())){  
 *family*.get(i).Laugh();  
 }  
 }  
 for (int i = 0; i < this.*family*.size(); i++){  
 if ((*family*.get(i).hashCode() > 1000) && (*family*.get(i).hashCode() < 2000) && (*family*.get(i).getActivity())){  
 *family*.get(i).Thoughts();  
 }  
 }  
 }  
 public static Human getMember(int i){  
 return *family*.get(i);  
 }  
 public static Human getNotMember(int i){  
 return *notfamily*.get(i);  
 }  
 public static int sizeFamily(){  
 return *family*.size();  
 }  
 public static int sizeNotFamily(){  
 return *notfamily*.size();  
 }  
  
}

1. Status

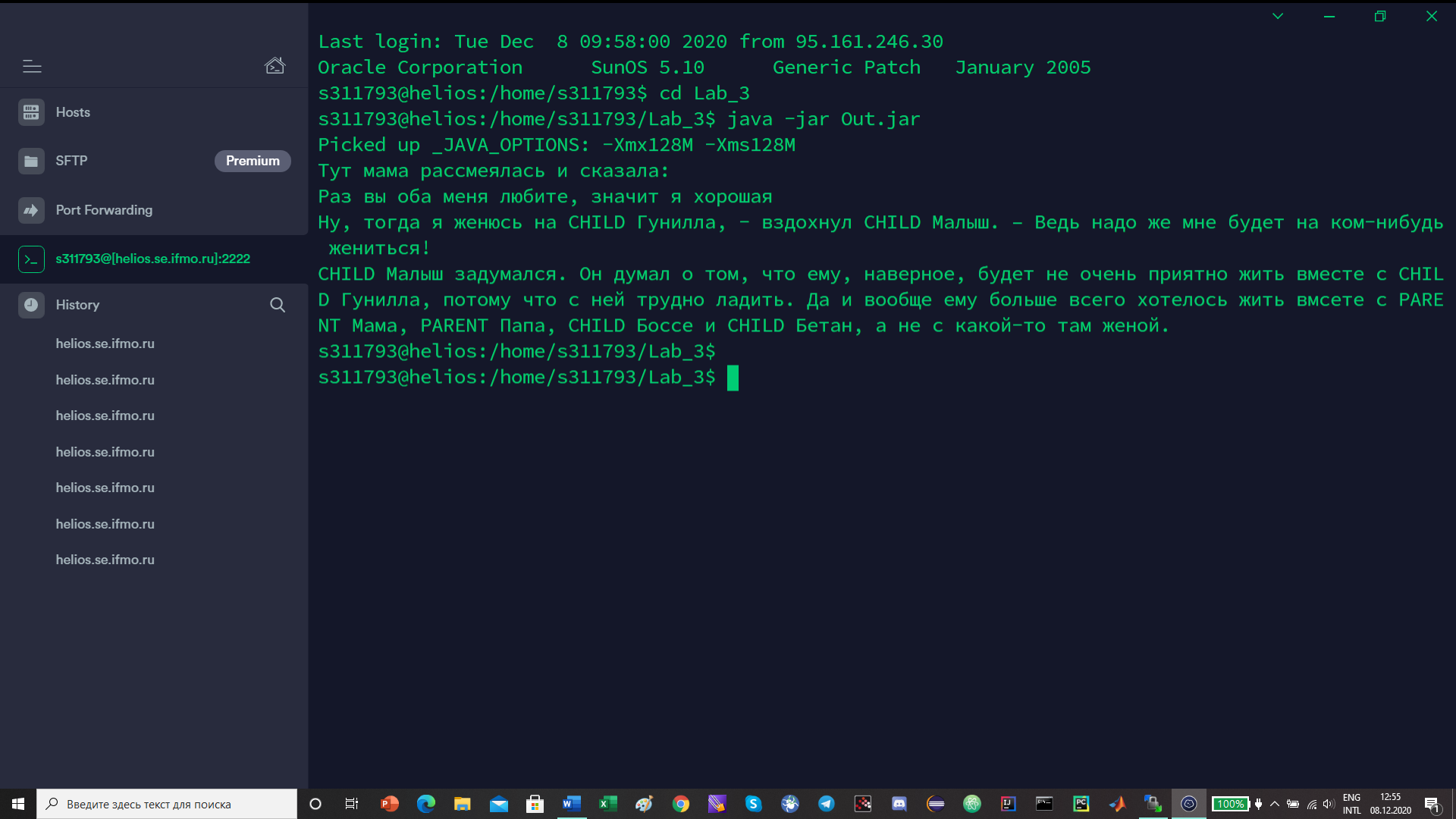
package juniorandkarlson;  
  
public enum Status {  
 *PARENT*,  
 *CHILD*,

DOG;  
  
 private Status() {  
 }  
}

1. Dog

package juniorandkarlson;  
  
public class Dog extends Human{  
  
  
 public Dog(String name){  
 super(name);  
 this.setGender(Gender.*ANIMAL*);  
 this.setStatus(Status.*DOG*);  
 }  
  
 @Override  
 public void Laugh() {  
  
 }  
  
 @Override  
 public void Thoughts() {  
  
 }  
}

**Результат работы программы:**



**Вывод:**

Изучены принципы объектно-ориентированного программирования SOLID и STUPID, особенности реализации наследования в Java, понятие абстрактного класса, перечисляемый тип данных (enum) в Java, перегрузка и переопределение методов