Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 10

по дисциплине ***«Объектно-ориентированные технологии программирования и стандарты проектирования»***

тема: «Основы Объектно-Ориентированного Программирования в Java. Абстракция. Инициализация объектов»

Название бригады: «TNNF»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студенты группы 10702217  Аристова Д.Д., Косякова Д.Д. |
| Преподаватель: |  | Иванченко Виктор Викторович |

2018-2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 10**

Основы Объектно-Ориентированного Программирования в Java. Абстракция. Инициализация объектов

Цель работы:

Научится решать проблемы с помощью объектно-ориентированного подхода

(выделять из предметной (доменной) области основные сущности (абстракции); проектировать и реализовывать на базе выделенной абстракции классы;

создавать на базе классов объекты и применять к ним различные способы

инициализации).

**Задание**

Необходимо решить задачу с использованием методологии ООП. Для чего

необходимо подобрать самостоятельно соответствующую проблемную

(предметную или доменную) область, которая базируется на объектах реального мира. Спроектировать классы (пользовательские типы данных) в Java для

программного представления данных объектов и основной логики системы.

Программа должна решать, как минимум, два полезных действия и иметь следующие вещи:

* не менее 3 разнообразных классов предметной области;
* не менее 5 атрибутов (характеристик) в каждом соответствующем классесущности;
* не менее 3 методов (поведения) в соответствующих функциональных

классах;

* хранить глобальные характеристики системы или характеристики уровня

отдельных классов;

* всевозможные средства инициализации (блоки инициализации, конструкторы по умолчанию, конструкторы с параметрами, конструкторы-копирования и т.д.) для объектов классов-сущностей.

Написать программу для создания объектов спроектированной системы и демонстрации взаимодействия между ними.

Дополнительно необходимо проанализировать стадии и способы инициализации состояния объекта и их очередность вызова JVM при создании соответствующих объектов. Отобразить результат и выводы в отчёте.

Требования к выполнению

1. Каждому участнику из команды нужно просмотреть все задания и выполнить по одному понравившемуся заданию из каждого раздела. Задания из соответствующих разделов внутри команды не должны повторяться.
2. Для некоторых вычислительных алгоритмов (решается самостоятельно) необходимо спроектировать блок-схему решения, которую необходимо поместить в отчёт или конспект.
3. Все алгоритмы должны быть решены с использованием итеративного подхода (однако, если есть желание, то можно также привести параллельно решение задания с использованием рекурсивного подхода).
4. На базе спроектированных алгоритмов разработать простейшее интерактивное консольное приложение с использование архитектурного шаблона проектирования ***Model-View-Controller****,* ***MVC***.
5. Создаваемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты: ***by.bntu.fitr.povt.nameofteam.javalabs.lab08****.*
6. При выполнения задания необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый и легко поддерживаемый и читаемый код.
7. Также рекомендуется придерживаться ***Single Responsibility Principle****,* ***SRP*** (принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода.
8. Если логически не подразумевается или в задании иного не указано, то входными и выходными данными являются вещественные числа (числа с плавающей запятой).
9. Все задания необходимо решать используя только базовые операции (простые операторы), определённые над примитивными типами данных в языке программирования Java, и условные и циклические конструкции (т.е. не нужно использовать массивы или любые другие контейнеры данных, операции над строковыми типами данных и т.д.).
10. В соответствующих компонентах бизнес-логики необходимо предусмотреть «защиту от дурака».
11. Для генерирования случайных чисел воспользуйтесь методами объекта класса ***java.util.Random***, а для реализации ввода данных с консоли (терминала) – соответствующими методами объекта класса ***java.util.Scanner***.
12. Программа должна обязательно быть снабжена комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчиков, название бригады (если есть), номер группы и дату разработки. Исходный текст классов и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.
13. Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем.
14. Интерфейс программы и комментарии должны быть на английском языке.
15. При проверки работоспособности приложения необходимо проверить все тестовые случаи.
16. При выполнении задания не рекомендуется использовать интегрированные средства разработки (*Integrated Development Environment, IDE*). Лучше задействовать любой текстовый редактор и основные компоненты Java (компилятор – ***javac***, утилиту для запуска *JVM* – ***java***).
17. При разработке программ придерживайтесь соглашений по написанию кода на *Java* (***Java Code-Convention****)* !!!

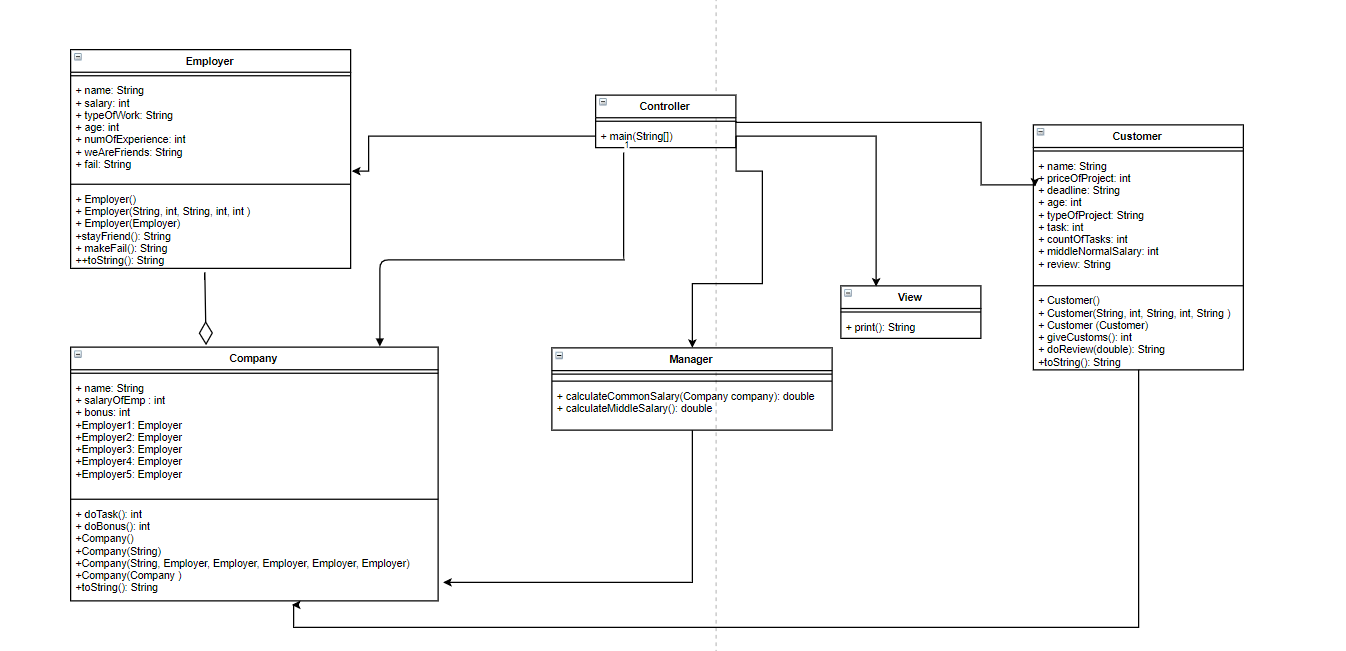


Рисунок 1 – UML – диаграмма классов

Результаты выполнения основного задания

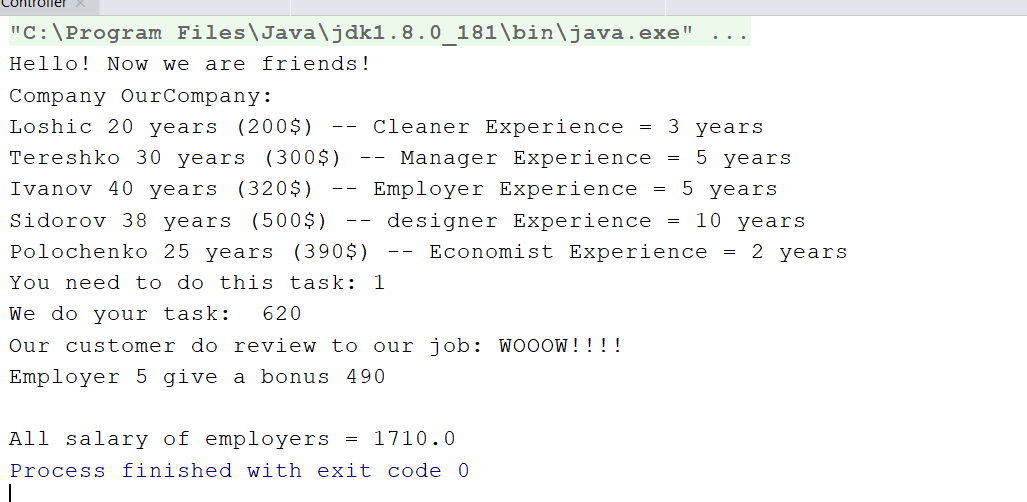


Рисунок 1 – результат работы основного задания

**Результаты выполнения дополнительного задания**

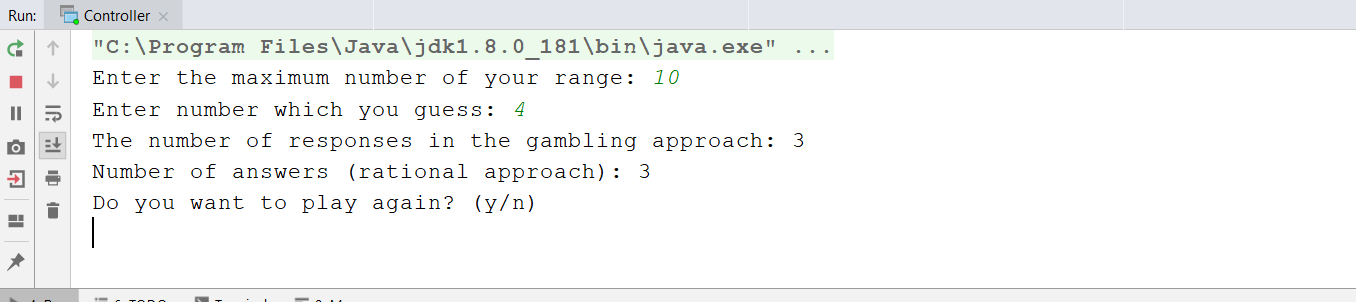


Рисунок 1 – результат работы дополнительного задания

Что мы узнали нового в процессе выполнения лабораторной работы (выводы)

1) Статическая переменная задействует память только один раз во время загрузки класса;

2) Можно создавать различные конструкторы, в зависимости от того, какие параметры будут передаваться;

3) Каждый класс может иметь также специальные методы, которые автоматически вызываются при создании и(или) уничтожении объектов класса:

4) Ключевое слово this стоит использовать:

* Когда у переменного экземпляра класса и переменной метода/конструктора одинаковые имена;
* Когда нужно вызвать конструктор одного типа (например, конструктор по умолчанию или параметризированный) из другого. Это еще называется явным вызовом конструктора.

5) @Override указывает, что далее мы собираемся переопределять метод базового класса.

6) если в базовом классе не окажется метода, то мы получим предупреждение компилятора о том, что хотя мы и собирались что-то переопределить, по факту этого не произошло.

*ПРИЛОЖЕНИЕ А*

Листинг исходного кода класса(ов)

Class Controller

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.controller;  
  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity.Company;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity.Customer;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity.Employer;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.logic.Manager;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.view.View;  
  
**public class** Controller {  
 **public static void** main(String[] args) {  
 Company company = **new** Company(**"OurCompany"**);  
 Customer customer = **new** Customer();  
 company.**employer1** = **new** Employer(**"Loshic"**, 200, **"Cleaner"**, 20, 3);  
 company.**employer2** = **new** Employer(**"Tereshko"**, 300, **"Manager"**, 30, 5);  
 company.**employer3** = **new** Employer(**"Ivanov"**, 320, **"Employer"**, 40, 5);  
 company.**employer4** = **new** Employer(**"Sidorov"**, 500, **"designer"**, 38, 10);  
 company.**employer5** = **new** Employer(**"Polochenko"**, 390, **"Economist"**, 25, 2);  
  
 **double** allSalary = Manager.*calculateCommonSalary*(company);  
 **double** middleSalary = Manager.*calculateMiddleSalary*();  
  
 View.*print*(company.**employer3**.stayFriend() + **"\n"**);  
 View.*print*(company + **""**);  
 **int** task = customer.giveCustoms();  
 View.*print*(**"You need to do this task: "** + task + **"\n"**);  
 company.**employer1**.makeFail();  
 View.*print*(**"We do your task: "** + company.doTask(task) + **"\n"**);  
 View.*print*(**"Our customer do review to our job: "** + customer.doReview(middleSalary) + **"\n"**);  
 View.*print*(**"Employer 5 give a bonus "** + company.doBonus(company.**employer5**) + **"\n"**);  
  
 View.*print*(**"\nAll salary of employers = "** + allSalary);  
 }  
}

Class Employer

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity;  
  
**public class** Employer {  
 **public static int** *employerAmount*;  
 **public** String **name**;  
 **public int salary**;  
 **public** String **typeOfWork**;  
 **public int age**;  
 **public int numOfExperience**;  
 **public** String **weAreFriends**;  
 **public** String **fail**;  
  
 **static** {  
 *employerAmount* = 0;  
 }  
  
 {  
 *employerAmount*++;  
 }  
  
  
 **public** Employer() {  
 **name** = **"no name"**;  
 **salary** = 100;  
 **typeOfWork** = **"no job"**;  
 **age** = 0;  
 **numOfExperience** = 0;  
 }  
  
 **public** Employer(String name, **int** salary, String typeOfWork, **int** age, **int** numOfExperience) {  
 **this**.**name** = name;  
 **this**.**salary** = salary;  
 **this**.**typeOfWork** = typeOfWork;  
 **this**.**age** = age;  
 **this**.**numOfExperience** = numOfExperience;  
 }  
  
 **public** Employer(Employer employer) {  
 **name** = employer.**name**;  
 **salary** = employer.**salary**;  
 **typeOfWork** = employer.**typeOfWork**;  
 **age** = employer.**age**;  
 **numOfExperience** = employer.**numOfExperience**;  
 }  
  
 **public** String makeFail() {  
 **fail** = **"UUUPPPS"**;  
 **return fail**;  
 }  
  
 **public** String stayFriend() {  
 **weAreFriends** = **"Hello! Now we are friends!"**;  
 **return weAreFriends**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return name** + **" "** + **age** + **" years "** + **"("** + **salary** + **"$"** + **")"** + **" -- "** + **typeOfWork** + **" "** + **"Experience = "** +  
 **numOfExperience** + **" "** + **"years"**;  
 }  
  
  
}

Class Company

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity;  
  
**public class** Company {  
 **public static final int *DEFAULT\_EMPOLOER\_AMOUNT*** = 5;  
  
 **public** String **name**;  
 **public** Employer **employer1**;  
 **public** Employer **employer2**;  
 **public** Employer **employer3**;  
 **public** Employer **employer4**;  
 **public** Employer **employer5**;  
 **public int salaryOfEmp**;  
 **public int bonus**;  
  
  
  
 **public** Company() {  
 }  
  
 **public** Company(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** Company(String name, Employer emp1, Employer emp2, Employer emp3, Employer emp4, Employer emp5) {  
 **this**.**name** = name;  
 **employer1** = emp1;  
 **employer2** = emp2;  
 **employer3** = emp3;  
 **employer4** = emp4;  
 **employer5** = emp5;  
 }  
  
 **public** Company(Company company) {  
 **name** = company.**name**;  
 **employer1** = company.**employer1**;  
 **employer2** = company.**employer2**;  
 **employer3** = company.**employer3**;  
 **employer4** = company.**employer4**;  
 **employer5** = company.**employer5**;  
 }  
  
  
 @Override  
 **public** String toString() {  
 **return "Company "** + **name** + **":\n"** + **employer1** + **"\n"** + **employer2** + **"\n"** + **employer3** + **"\n"** + **employer4** + **"\n"** +  
 **employer5** + **"\n"**;  
 }  
  
 **public int** doTask(**int** task){  
 **switch**(task){  
 **case** 0:  
 **salaryOfEmp** = **employer1**.**salary** + **employer2**.**salary**;  
 **break**;  
 **case** 1:  
 **salaryOfEmp** = **employer3**.**salary** + **employer2**.**salary**;  
 **break**;  
 **case** 2:  
 **salaryOfEmp** = **employer4**.**salary** + **employer5**.**salary**;  
 }  
 **return salaryOfEmp**;  
 }  
 **public int** doBonus(Employer employer){  
 **return** employer.**salary** + 100;  
 }  
}

Class Customer

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity;  
  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.logic.Manager;  
  
**import** java.util.Random;  
  
**public class** Customer {  
 **public static int** *customsAmount*;  
 **public** String **name**;  
 **public int priceOfProject**;  
 **public** String **deadline**;  
 **public int age**;  
 **public** String **typeOfProject**;  
 **public int task**;  
 **public int countOfTasks** = 3;  
 Random **random** = **new** Random();  
 **public int middleNormalSalary** = 340;  
 **public** String **review**;  
  
 **static** {  
 *customsAmount* = 0;  
 }  
  
 {  
 *customsAmount*++;  
 }  
  
 **public** Customer() {  
 **name** = **"no name"**;  
 **priceOfProject** = 300;  
 **deadline** = **"no limit"**;  
 **age** = 0;  
 **typeOfProject** = **"No project"**;  
 }  
  
 **public** Customer(String name, **int** priceOfProject, String deadline, **int** age, String typeOfProject) {  
 **this**.**name** = name;  
 **this**.**priceOfProject** = priceOfProject;  
 **this**.**deadline** = deadline;  
 **this**.**age** = age;  
 **this**.**typeOfProject** = typeOfProject;  
 }  
  
 **public** Customer(Customer customer) {  
 **name** = customer.**name**;  
 **priceOfProject** = customer.**priceOfProject**;  
 **deadline** = customer.**deadline**;  
 **age** = customer.**age**;  
 **typeOfProject** = customer.**typeOfProject**;  
 }  
  
 **public int** giveCustoms(){  
 **task** = **random**.nextInt(**countOfTasks**);  
 **return task**;  
 }  
 **public** String doReview(**double** middleSalary){  
 **if** (middleSalary>=**middleNormalSalary**){  
 **review** = **"WOOOW!!!!"** ;  
 }  
 **else**{  
 **review** = **"OH MY GOD!!! YOU'RE STUIPID COMPANY!!!!"**;  
 }  
 **return review**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return name** + **" "** + **age** + **" years "** + **"("** + **priceOfProject** + **"$"** + **")"** + **" -- "** + **deadline** + **" "** + **" Type of project: "** +  
 **typeOfProject**;  
 }  
}

Class Manager

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.logic;  
  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity.Company;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.entity.Customer;  
  
**public class** Manager {  
 **public static int** *allSalary*;  
 **public static double** calculateCommonSalary(Company company){  
 *allSalary* = company.**employer1**.**salary** + company.**employer2**.**salary** + company.**employer3**.**salary** + company.**employer4**.**salary** + company.**employer5**.**salary**;  
 **return** *allSalary*;  
 }  
 **public static double** calculateMiddleSalary(){  
 **double** middleSalary = *allSalary*/2;  
 **return** middleSalary;  
 }  
}

Class View

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.view;  
  
**public class** View {  
 **public static void** print(String msg) {  
 System.***out***.print(msg);  
 }  
}

Дополнительное задание

Class Controller

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.Controller;  
  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.Logic.Logic;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.Tool.Check;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.Tool.UserInput;  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.View.View;  
  
**public class** Controller {  
 **public static void** main(String[] args) {  
 *additionalTask*();  
 }  
  
 **private static void** additionalTask()  
 {  
 **boolean** isRun = **true**;  
 **while** (isRun)  
 {  
 **int** maxNmber = UserInput.*inputInt*(**"Enter the maximum number of your range: "**);  
 **int** guessNumb = UserInput.*inputInt*(**"Enter number which you guess: "**);  
  
 View.*print*(String.*format*(  
 **"The number of responses in the gambling approach: %d\n"** +  
 **"Number of answers (rational approach): %d"**,  
 Logic.*excitementGuess*(maxNmber, guessNumb), Logic.*binaryGuess*(guessNumb, maxNmber)));  
  
 **char** exit = UserInput.*inputChar*(**"\nDo you want to play again? (y/n)\n"**);  
 isRun = Check.*checkEndRuning*(exit);  
 }  
 }  
  
}

Class Check

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.Tool;  
  
**public class** Check {  
 **public static boolean** checkEndRuning(**char** answer)  
 {  
 **boolean** isRun = **true**;  
  
 **if** (answer == **'N'** || answer == **'n'**) {  
 isRun = **false**;  
 **return** isRun;  
 }  
  
 **return** isRun;  
 }  
}

Class UserInput

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.Tool;  
  
**import** by.bntu.fitr.povt.tnnf.javalabs.lab10.View.View;  
**import** java.util.Scanner;  
  
**public class** UserInput {  
 **private static** Scanner *in* = **new** Scanner(System.***in***);  
  
 **public static float** inputFloat(String msg)  
 {  
 View.*print*(msg);  
 **return** *in*.nextFloat();  
 }  
  
 **public static int** inputInt(String msg)  
 {  
 View.*print*(msg);  
 **return** *in*.nextInt();  
 }  
  
 **public static char** inputChar(String msg)  
 {  
 View.*print*(msg);  
 **return** *in*.next().charAt(0);  
 }  
}

Class Logic

**package** by.bntu.fitr.povt.tnnf.javalabs.lab10.Logic;  
  
**import** java.util.Random;  
  
**public class** Logic {  
  
 **private static** Random *rnd* = **new** Random(System.*currentTimeMillis*());  
 *// Additional Task* **public static int** excitementGuess(**int** max, **int** guessNum)  
 {  
 **int** answer = *rnd*.nextInt(max + 1);  
 **int** countAnswer = 0;  
 **while** (answer != guessNum)  
 {  
 countAnswer++;  
 **if** (answer > guessNum)  
 {  
 **int** i = answer;  
 answer = *rnd*.nextInt(i);  
 }  
 **else** {  
 **int** diff = guessNum - answer + 1;  
 **int** min = answer;  
 answer = *rnd*.nextInt(diff) + min;  
 }  
 }  
 **return** countAnswer;  
 }  
  
 **public static int** binaryGuess(**int** guessNumb, **int** max)  
 {  
 **int** countAnswer = 0;  
 **int** min = 0;  
 **int** answer = (min + max)/2;  
 **while** (answer != guessNumb) {  
 answer = (min + max) / 2;  
 **if** (answer > guessNumb) {  
 countAnswer++;  
 max = answer;  
 } **else if** (answer < guessNumb) {  
 countAnswer++;  
 min = answer;  
 }  
 }  
 **return** countAnswer;  
 }  
  
}