Міністерство освіти і науки України

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

**Звіт**

з лабораторних робіт

з дисципліни «Розробка ПЗ мовою Java»

Виконав студент 3-го курсу, групи ВТ-22-2

спеціальності 121 «Інженерії програмного забезпечення»

Т.Ю. Регенель

Керівник В.І. Піонтківський

Житомир – 2024

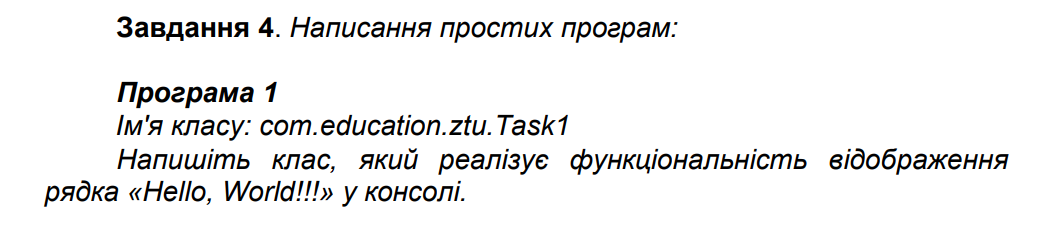
ЗМІСТ

[Лабораторна робота №1 3](#_Toc185618407)

[Лабораторна робота №2 8](#_Toc185618408)

[Лабораторна робота №3 32](#_Toc185618409)

# Лабораторна робота №1



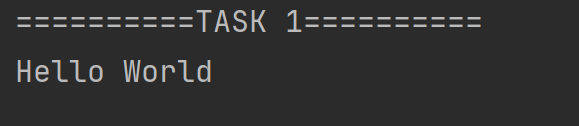
public class Task1 {

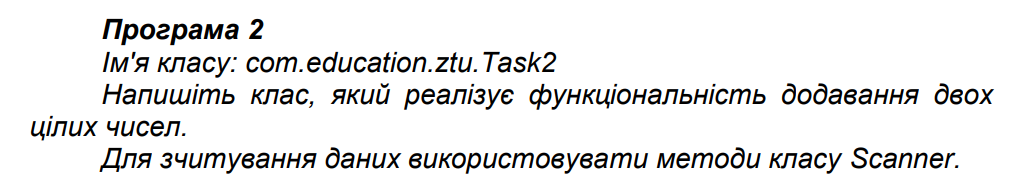
public static void main(String[] args) {

System.out.println("Hello World");

}

}





public class Task2 {

public static void main(String[] args, Scanner sc) {

System.out.print("a: ");

int a = sc.nextInt();

System.out.print("b: ");

int b = sc.nextInt();

System.out.println("a + b = " + Sum(a, b));

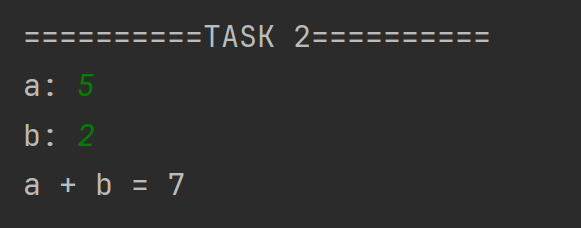
}

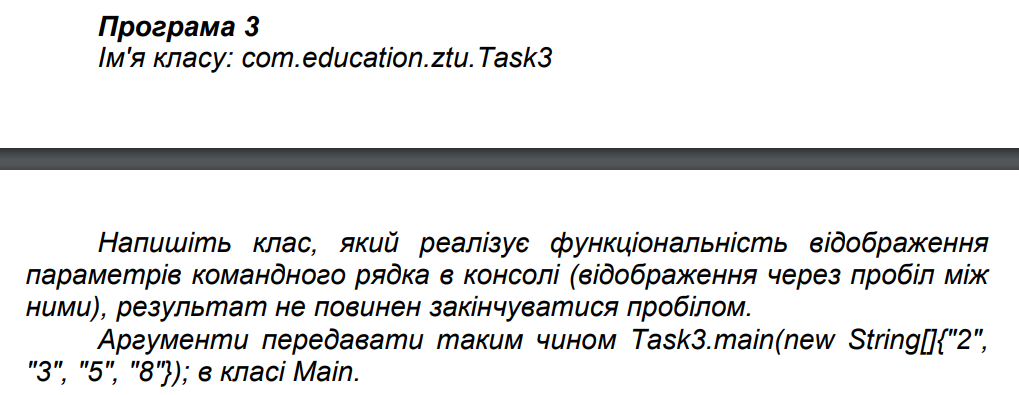
public static int Sum(int a, int b) {

return a + b;

}

}





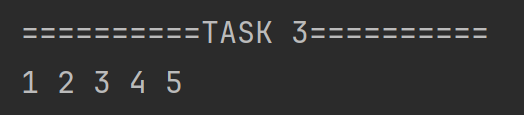
public class Task3 {

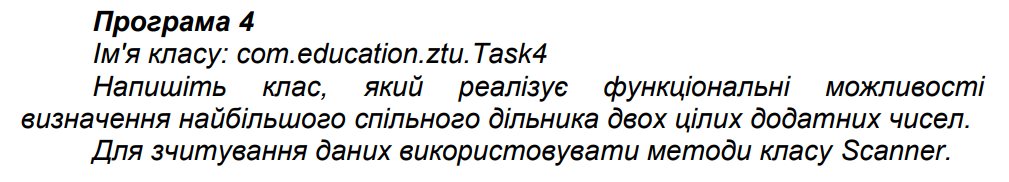
public static void main(String[] args) {

System.out.println(String.join(" ", args));

}

}





public class Task4 {

public static void main(String[] args, Scanner sc) {

System.out.print("a: ");

int a = sc.nextInt();

System.out.print("b: ");

int b = sc.nextInt();

System.out.println("MCM for a and b: " + GetMCM(a, b));

}

private static int GetMCM(int a, int b) {

while (a != 0 && b != 0) {

if (a > b) {

a = a % b;

} else {

b = b % a;

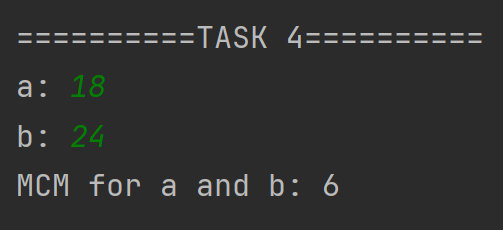
}

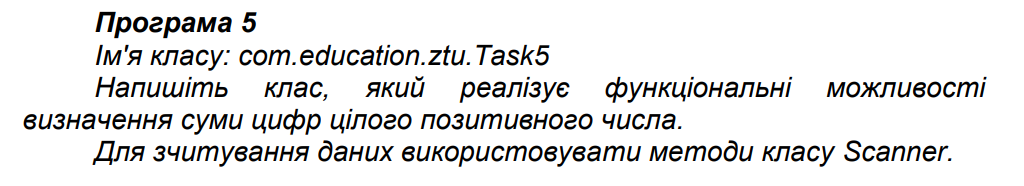
}

return a + b;

}

}





public class Task5 {

public static void main(String[] args, Scanner sc) {

System.out.print("n: ");

int n = sc.nextInt();

System.out.println("Digits sum for n: " + GetDigitsSum(n));

}

public static int GetDigitsSum(int a) {

int result = 0;

while (a > 9) {

result += a % 10;

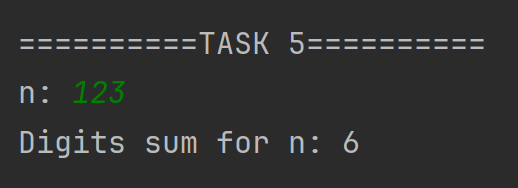
a = a / 10;

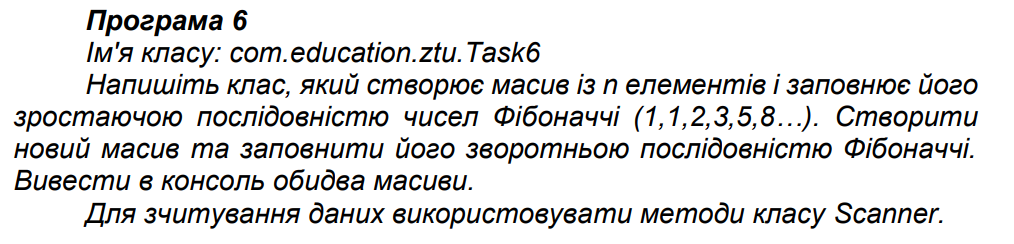
}

return result + a;

}

}





public class Task6 {

public static List<Integer> fibs = new ArrayList<Integer>(){{

add(1); add(1);

}};

public static void main(String[] args, Scanner sc) {

System.out.print("n: ");

int n = sc.nextInt();

List<Integer> rsultFibs = new ArrayList<>();

for(int i = 1; i <= n; i += 1) {

rsultFibs.add(Fib(i));

}

System.out.println("Fib for n: " + Fib(n));

System.out.println("Fibs list for n: " + rsultFibs.toString());

System.out.println("Fibs reversed list for n: " + rsultFibs.reversed().toString());

}

public static int Fib(int n){

if(fibs.size() >= n){

return fibs.get(n - 1);

}

for(int i = fibs.size(); i < n; i += 1){

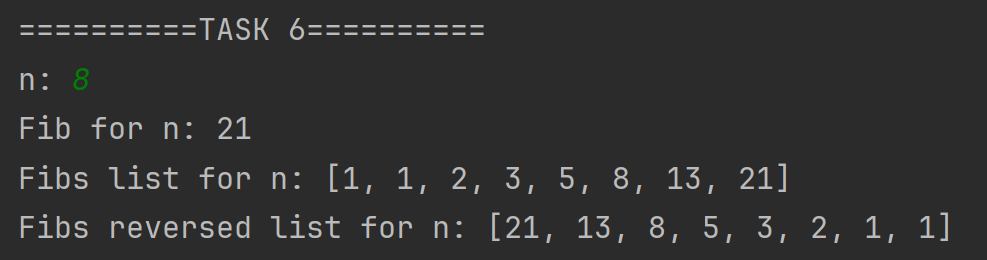
fibs.add(fibs.get(i - 1) + fibs.get(i - 2));

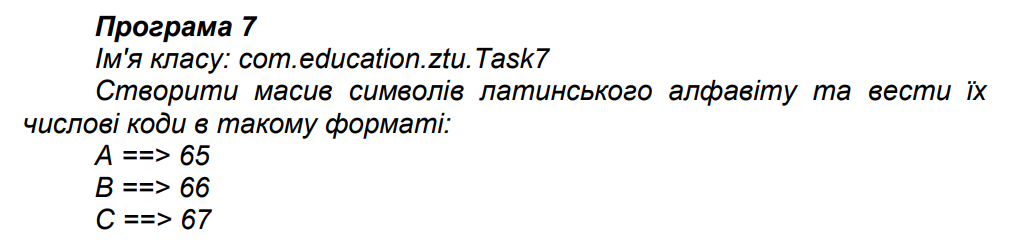
}

return fibs.get(n - 1);

}

}





public class Task7 {

public static void main(String[] args) {

String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

for (int i = 0; i < alphabet.length(); i += 1) {

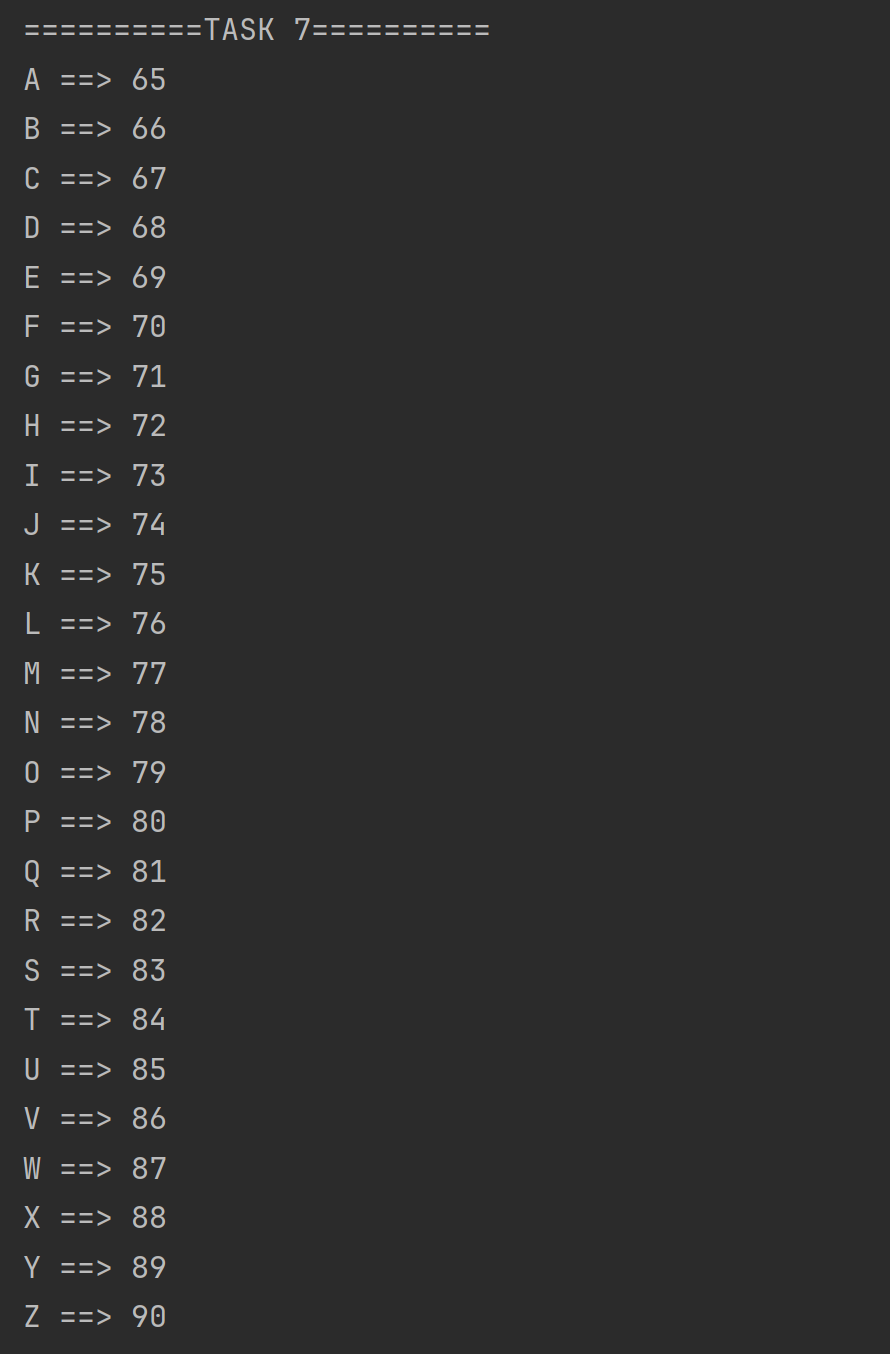
int code = (int)alphabet.charAt(i);

System.out.println(alphabet.charAt(i) + " ==> " + code);

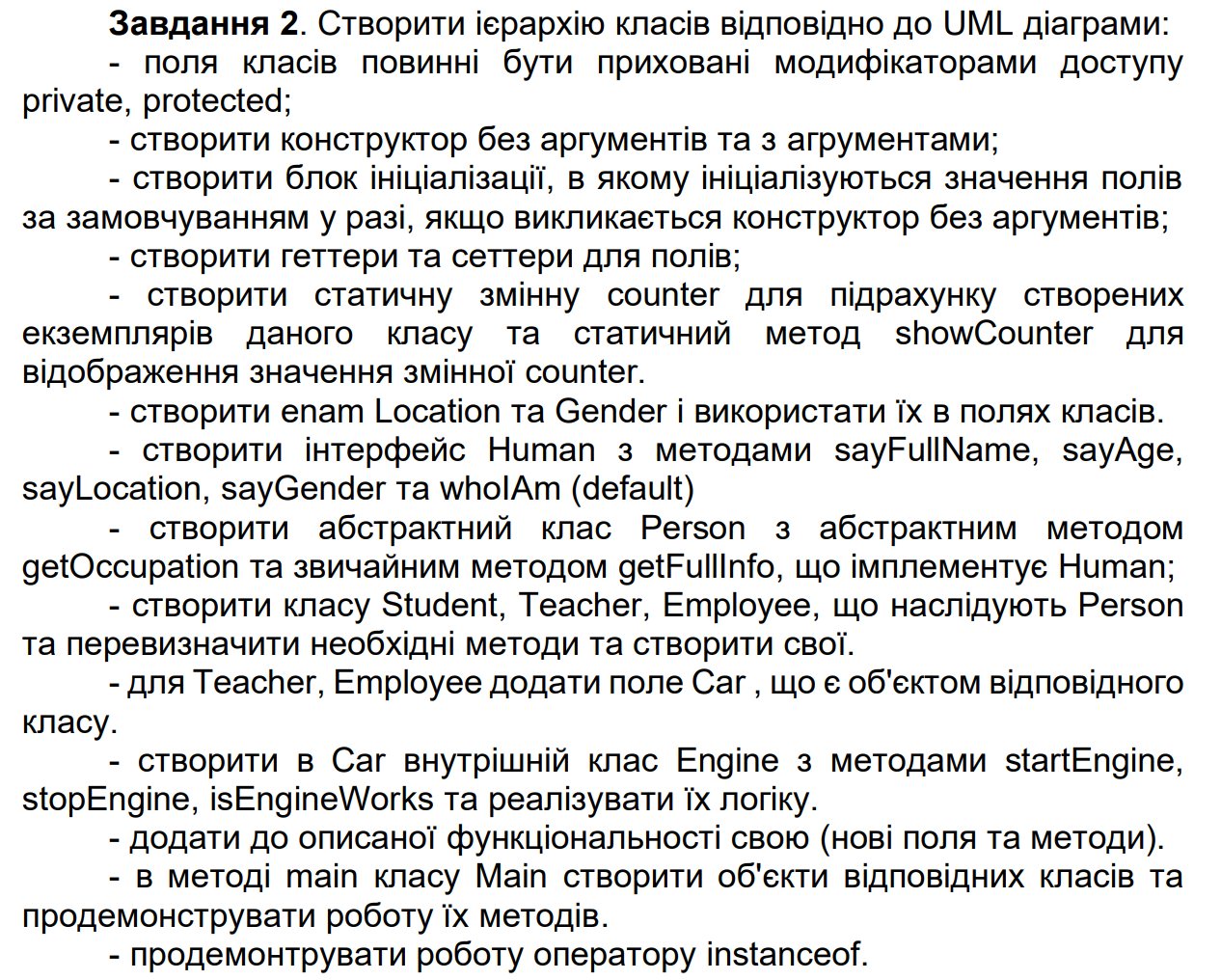
}

}

}



# Лабораторна робота №2



public interface Human {

void sayAge();

void sayGender();

void sayLocation();

void sayName();

default void whoIAm(){

System.out.println("I'm a human");

};

String getFullInfo();

}

// Why not a Boolean? 🤨

public enum Gender {

MALE,

FEMALE

}

public enum Location {

Alchevsk,

Almazna,

Alupka,

Alushta,

Amvrosiivka,

Ananiv,

Andrushivka,

Antratsyt,

Apostolove,

Armiansk,

Artsyz,

Avdiivka,

Bahacheve,

Bakhchysarai,

Bakhmach,

Bakhmut,

Balakliia,

Balta,

Bar,

Baranivka,

Barvinkove,

Bashtanka,

Baturyn,

Belz,

Berdiansk,

Berdychiv,

Berehove,

Berestechko,

Berestyn,

Berezan,

Berezhany,

Berezivka,

Berezne,

Bershad,

Beryslav,

Bibrka,

BilaTserkva,

BilhorodDnistrovskyi,

Biliaivka,

Bilohirsk,

Bilopillia,

Bilozerske,

Bilytske,

Blahovishchenske,

Bobrovytsia,

Bobrynets,

Bohodukhiv,

Bohuslav,

Boiarka,

BokovoKhrustalne,

Bolekhiv,

Bolhrad,

Borshchiv,

Boryslav,

Boryspil,

Borzna,

Brianka,

Brody,

Brovary,

Bucha,

Buchach,

Bunhe,

Burshtyn,

Buryn,

Busk,

ChasivYar,

Cherkasy,

Chernihiv,

Chernivtsi,

Chernobyl,

Chop,

Chornomorsk,

Chortkiv,

Chudniv,

Chuhuiv,

Chyhyryn,

Chystiakove,

Debaltseve,

Derazhnia,

Derhachi,

Dnipro,

Dniprorudne,

Dobromyl,

Dobropillia,

Dokuchaievsk,

Dolyna,

Dolynska,

Donetsk,

Dovzhansk,

Drohobych,

Druzhkivka,

Dubliany,

Dubno,

Dubrovytsia,

Dunaivtsi,

Dzhankoi,

Enerhodar,

Fastiv,

Feodosia,

Hadiach,

Haisyn,

Haivoron,

Halych,

Henichesk,

Hertsa,

Hirnyk,

Hirske,

Hlobyne,

Hlukhiv,

Hlyniany,

Hnivan,

HolaPrystan,

Holubivka,

HorishniPlavni,

Horlivka,

Horodenka,

Horodnia,

Horodok,

Horodyshche,

Horokhiv,

Hrebinka,

Huliaipole,

Ichnia,

Illintsi,

Ilovaisk,

Inkerman,

Irmino,

Irpin,

Irshava,

IvanoFrankivsk,

Iziaslav,

Izium,

Izmail,

Kadiivka,

Kaharlyk,

Kakhovka,

Kalmiuske,

Kalush,

Kalynivka,

KamianetsPodilskyi,

Kamianka,

KamiankaBuzka,

KamiankaDniprovska,

Kamianske,

KaminKashyrskyi,

Kaniv,

Karlivka,

Kerch,

Kharkiv,

Khartsyzk,

Kherson,

Khmelnytskyi,

Khmilnyk,

Khodoriv,

Khorol,

Khorostkiv,

Khotyn,

Khrestivka,

Khrustalnyi,

Khrystynivka,

Khust,

KhutirMykhailivskyi,

Khyriv,

Kiliia,

Kitsman,

Kivertsi,

Kobeliaky,

Kodyma,

Kolomyia,

Komarno,

Konotop,

Kopychyntsi,

Korets,

Koriukivka,

Korosten,

Korostyshiv,

KorsunShevchenkivskyi,

Kosiv,

Kostiantynivka,

Kostopil,

Kovel,

Koziatyn,

Kramatorsk,

Krasnohorivka,

Krasyliv,

Kremenchuk,

Kremenets,

Kreminna,

Krolevets,

Kropyvnytskyi,

KryvyiRih,

Kupiansk,

Kurakhove,

Kyiv,

Kypuche,

Ladyzhyn,

Lanivtsi,

Lebedyn,

Liuboml,

Liubotyn,

Lokhvytsia,

Lozova,

Lubny,

Luhansk,

Lutsk,

Lutuhyne,

Lviv,

Lyman,

Lypovets,

Lysychansk,

Makiivka,

MalaVyska,

Malyn,

Marhanets,

Marinka,

Mariupol,

Melitopol,

Mena,

Merefa,

Miusynsk,

MohylivPodilskyi,

Molochansk,

Molodohvardiisk,

Monastyryshche,

Monastyryska,

Morshyn,

Mospyne,

Mostyska,

Mukachevo,

Mykolaiv,

Mykolaivka,

Myrhorod,

Myrnohrad,

Myronivka,

Nadvirna,

Nemyriv,

Netishyn,

Nikopol,

Nizhyn,

Nosivka,

NovaKakhovka,

NovaOdesa,

NovhorodSiverskyi,

Novoazovsk,

Novodnistrovsk,

Novodruzhesk,

Novohrodivka,

Novoiavorivsk,

Novomyrhorod,

Novoselytsia,

Novoukrainka,

Novovolynsk,

NovyiBuh,

NovyiKalyniv,

NovyiRozdil,

Obukhiv,

Ochakiv,

Odesa,

Okhtyrka,

Oleksandriia,

Oleksandrivsk,

Oleshky,

Olevsk,

Orikhiv,

Oster,

Ostroh,

Ovruch,

Pavlohrad,

Perechyn,

Pereiaslav,

Peremyshliany,

Pereshchepyne,

Perevalsk,

Pervomaisk,

PetrovoKrasnosillia,

Piatykhatky,

Pidhaitsi,

Pidhorodne,

Pivdenne,

Pochaiv,

Podilsk,

Pohrebyshche,

Pokrov,

Pokrovsk,

Polohy,

Polonne,

Poltava,

Pomichna,

Popasna,

Pripyat,

Pryluky,

Prymorsk,

Pryvillia,

Pustomyty,

Putyvl,

Pyriatyn,

Radekhiv,

Radomyshl,

Radyvyliv,

Rakhiv,

RavaRuska,

Reni,

Reshetylivka,

Rivne,

Rodynske,

Rohatyn,

Romny,

Rovenky,

Rozdilna,

Rozhyshche,

Rubizhne,

Rudky,

Rzhyshchiv,

Saky,

Samar,

Sambir,

Sarny,

Selydove,

Semenivka,

SeredynaBuda,

Sevastopol,

Shakhtarsk,

Shakhtarske,

Sharhorod,

Shchastia,

Shcholkine,

Shepetivka,

Sheptytskyi,

Shostka,

Shpola,

Shumsk,

Sievierodonetsk,

Simferopol,

Siversk,

Skadovsk,

Skalat,

Skole,

Skvyra,

Slavuta,

Slavutych,

Sloviansk,

Smila,

Sniatyn,

Snihurivka,

Snizhne,

Snovsk,

Sokal,

Sokyriany,

Soledar,

Sorokyne,

Sosnivka,

Starobilsk,

Starokostiantyniv,

StaryiKrym,

StaryiSambir,

Stebnyk,

Storozhynets,

Stryi,

Sudak,

SudovaVyshnia,

Sukhodilsk,

Sumy,

Svaliava,

Svatove,

Sviatohirsk,

Svitlodarsk,

Svitlovodsk,

Synelnykove,

Talne,

Tarashcha,

Tatarbunary,

Tavriisk,

Teplodar,

Terebovlia,

Ternivka,

Ternopil,

Tetiiv,

Tiachiv,

Tlumach,

Tokmak,

Toretsk,

Trostianets,

Truskavets,

Tulchyn,

Turka,

Tysmenytsia,

Uhniv,

Ukrainka,

Ukrainsk,

Uman,

Ustyluh,

Uzhhorod,

Uzyn,

Valky,

Varash,

Vashkivtsi,

Vasylivka,

Vasylkiv,

VelykiMosty,

Verkhivtseve,

Verkhniodniprovsk,

Vilniansk,

Vilnohirsk,

Vinnytsia,

Volnovakha,

Volochysk,

Volodymyr,

Vorozhba,

Vovchansk,

Voznesenivka,

Voznesensk,

Vuhledar,

Vuhlehirsk,

Vylkove,

Vynnyky,

Vynohradiv,

Vyshhorod,

Vyshneve,

Vyzhnytsia,

Yahotyn,

Yalta,

Yampil,

YanyKapu,

Yaremche,

Yasynuvata,

Yavoriv,

Yenakiieve,

Yevpatoria,

Yuzhne,

Yuzhnoukrainsk,

Zalishchyky,

Zalizne,

Zaporizhzhia,

Zastavna,

Zavodske,

Zbarazh,

Zboriv,

Zdolbuniv,

Zelenodolsk,

Zhashkiv,

Zhdanivka,

Zhmerynka,

Zhovkva,

ZhovtiVody,

Zhydachiv,

Zhytomyr,

Zinkiv,

Zlatopil,

Zmiiv,

Znamianka,

Zolochiv,

Zolote,

Zolotonosha,

Zorynsk,

Zuhres,

Zvenyhorodka,

Zviahel,

Zymohiria

}

public abstract class Person implements Human {

protected int age;

protected String firstName;

protected String lastName;

protected Gender gender;

protected Location location;

{

age = 0;

firstName = "NoName";

lastName = "NoName";

gender = Gender.MALE;

location = Location.Zhytomyr;

}

public Person() {

}

public Person(int age, String firstName, String lastName, Gender gender, Location location) {

this.age = age;

this.firstName = firstName;

this.lastName = lastName;

this.gender = gender;

this.location = location;

}

public abstract void getOccupation();

public void sayAge() {

System.out.println("Age: " + age);

}

public void sayGender() {

System.out.println("Gender: " + gender);

}

public void sayLocation() {

System.out.println("Location: " + location);

}

public void sayName() {

System.out.println("Name: " + firstName + " " + lastName);

}

@Override

public void whoIAm() {

System.out.println("I'm a person");

}

@Override

public String getFullInfo() {

StringBuilder sb = new StringBuilder(this.getClass().getSimpleName() + ": ");

sb.append(firstName + " " + lastName);

sb.append(", " + gender);

sb.append(", " + age + " y.o.");

sb.append(", lives in " + location);

return sb.toString();

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getFullName(){

return firstName + " " + lastName;

}

public Gender getGender() {

return gender;

}

public void setGender(Gender gender) {

this.gender = gender;

}

public Location getLocation() {

return location;

}

public void setLocation(Location location) {

this.location = location;

}

public String toString(){

return getFullName();

}

}

public class Student extends Person {

private static int instanceCount = 0;

public static void showInstanceCount(){

System.out.println("Teacher instance count: " + instanceCount);

};

private int course;

private String specialty;

private String university;

public Student(int course, String specialty, String university) {

this.course = course;

this.specialty = specialty;

this.university = university;

instanceCount += 1;

}

public Student(int age, String firstName, String lastName, Gender gender, Location location, int course, String specialty, String university) {

super(age, firstName, lastName, gender, location);

this.course = course;

this.specialty = specialty;

this.university = university;

instanceCount += 1;

}

@Override

public void getOccupation() {

System.out.println("Occupation: studying");

}

@Override

public String getFullInfo(){

StringBuilder sb = new StringBuilder(super.getFullInfo());

sb.append(", studying at " + university);

sb.append(", on " + specialty + " speciality");

sb.append(", " + course + " course");

return sb.toString();

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public String getSpecialty() {

return specialty;

}

public void setSpecialty(String specialty) {

this.specialty = specialty;

}

public String getUniversity() {

return university;

}

public void setUniversity(String university) {

this.university = university;

}

}

public class Teacher extends Person {

private static int instanceCount = 0;

public static void showInstanceCount(){

System.out.println("Teacher instance count: " + instanceCount);

};

private Car car;

private String subject;

private String university;

public Teacher(Car car, String subject, String university) {

this.car = car;

this.subject = subject;

this.university = university;

instanceCount += 1;

}

public Teacher(int age, String firstName, String lastName, Gender gender, Location location, Car car, String subject, String university) {

super(age, firstName, lastName, gender, location);

this.car = car;

this.subject = subject;

this.university = university;

instanceCount += 1;

}

@Override

public void getOccupation() {

System.out.println("Occupation: teaching");

}

@Override

public String getFullInfo(){

StringBuilder sb = new StringBuilder(super.getFullInfo());

sb.append(", works in " + university);

sb.append(", teaches " + subject);

sb.append(", drives " + car.getBrand());

return sb.toString();

}

public Car getCar() {

return car;

}

public void setCar(Car car) {

this.car = car;

}

public String getSubject() {

return subject;

}

public void setSubject(String subject) {

this.subject = subject;

}

public String getUniversity() {

return university;

}

public void setUniversity(String university) {

this.university = university;

}

}

public class Employee extends Person {

private static int instanceCount = 0;

public static void showInstanceCount(){

System.out.println("Teacher instance count: " + instanceCount);

};

private Car car;

private String company;

private String position;

public Employee(Car car, String company, String position) {

this.car = car;

this.company = company;

this.position = position;

instanceCount += 1;

}

public Employee(int age, String firstName, String lastName, Gender gender, Location location, Car car, String company, String position) {

super(age, firstName, lastName, gender, location);

this.car = car;

this.company = company;

this.position = position;

instanceCount += 1;

}

@Override

public void getOccupation() {

System.out.println("Occupation: working");

}

@Override

public String getFullInfo(){

StringBuilder sb = new StringBuilder(super.getFullInfo());

sb.append(", works in " + company + " as " + position);

sb.append(", drives " + car.getBrand());

return sb.toString();

}

public String getCompany() {

return company;

}

public void setCompany(String company) {

this.company = company;

}

public String getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

}

public class Car {

public class Engine {

private boolean working;

public Engine() {

this.working = false;

}

public void startEngine(){

this.working = true;

}

public void stopEngine(){

this.working = false;

}

public boolean isWorking() {

return working;

}

}

private String brand;

private Engine engine;

public Car(String brand) {

this.brand = brand;

this.engine = new Engine();

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

public Engine getEngine() {

return engine;

}

public boolean isEngineRunning(){

return engine.isWorking();

}

}

public class Bank {

private Map<Person, Double> accounts;

private String name;

public Bank(String name) {

this.accounts = new HashMap<Person, Double>();

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

log("This bank is called \"" + name + "\" now");

this.name = name;

}

public void openAccount(Person person) {

this.accounts.put(person, 0.0);

log(person + " opened an account");

}

public void deposit(Person person, Double amount) {

throwIfNoAccount(person);

throwIfBadAmount(amount);

accounts.put(person, accounts.get(person) + amount);

log(person + " deposit " + amount + " UAH");

}

public void withdraw(Person person, Double amount) {

throwIfNoAccount(person);

throwIfBadAmount(person, amount);

accounts.put(person, accounts.get(person) - amount);

log(person + " withdraw " + amount + " UAH");

}

public void transfer(Person sender, Person receiver, Double amount) {

throwIfNoAccount(sender);

throwIfNoAccount(receiver);

throwIfBadAmount(sender, amount);

accounts.put(sender, accounts.get(sender) - amount);

accounts.put(receiver, accounts.get(receiver) + amount);

log(sender + " send " + amount + " UAH to " + receiver);

}

private void throwIfNoAccount(Person person) {

if(!accounts.containsKey(person)) {

throw new IllegalArgumentException(person + " does not have an account in this bank");

}

}

private void throwIfBadAmount(double amount) {

if(amount < 0){

throw new IllegalArgumentException("Amount must be greater than 0");

}

}

private void throwIfBadAmount(Person person, double amount) {

throwIfBadAmount(amount);

throwIfNotEnoughMoney(person, amount);

}

private void throwIfNotEnoughMoney(Person person, Double amount) {

if(amount > accounts.get(person)) {

throw new IllegalArgumentException("There is not enough money on the " + person + "'s account");

}

}

private void log(String message){

System.out.println("[" + name + "] " + message);

}

}

public class Main {

public static void main(String[] args) {

Bank mono = new Bank("Mono");

Car AM5203AA = new Car("VolksWagen");

Teacher mrWise = new Teacher(

37,

"John",

"Wise",

Gender.MALE,

Location.Zhytomyr,

AM5203AA,

"Docker",

"ZTU"

);

mono.openAccount(mrWise);

Student max = new Student(

20,

"Max",

"Adamson",

Gender.MALE,

Location.Zhytomyr,

3,

"Software Engineering",

"ZTU"

);

mono.openAccount(max);

Student emily = new Student(

19,

"Emily",

"Roberts",

Gender.FEMALE,

Location.Zhytomyr,

3,

"Software Engineering",

"ZTU"

);

mono.openAccount(emily);

Student sam = new Student(

19,

"Samuel",

"Price",

Gender.MALE,

Location.Zhytomyr,

3,

"Software Engineering",

"ZTU"

);

mono.openAccount(sam);

Car AM8639AB = new Car("Volvo");

Employee employedDude = new Employee(

27,

"Dude",

"Dudson",

Gender.MALE,

Location.Zhytomyr,

AM8639AB,

"Employed Dudes inc.",

"Worker"

);

mono.openAccount(employedDude);

mono.deposit(employedDude, 25350.00);

System.out.println("Society:");

System.out.println(mrWise.getFullInfo());

mrWise.getOccupation();

System.out.println(max.getFullInfo());

max.getOccupation();

System.out.println(emily.getFullInfo());

emily.getOccupation();

System.out.println(sam.getFullInfo());

sam.getOccupation();

System.out.println(employedDude.getFullInfo());

employedDude.getOccupation();

System.out.println("");

mono.transfer(employedDude, sam, 180.00);

System.out.println("");

checkInstance(mrWise, Teacher.class);

checkInstance(mrWise, Person.class);

checkInstance(mrWise, Human.class);

checkInstance(mrWise, Student.class);

System.out.println("");

System.out.println("Available locations: " + Arrays.toString(Location.values()).replace("[", "").replace("]", ""));

}

public static void checkInstance(Object instance, Class<?> of)

{

String result = isInstanceOf(instance, of) ? "is" : "is not";

System.out.println(instance + " " + result + " instance of " + of.getSimpleName());

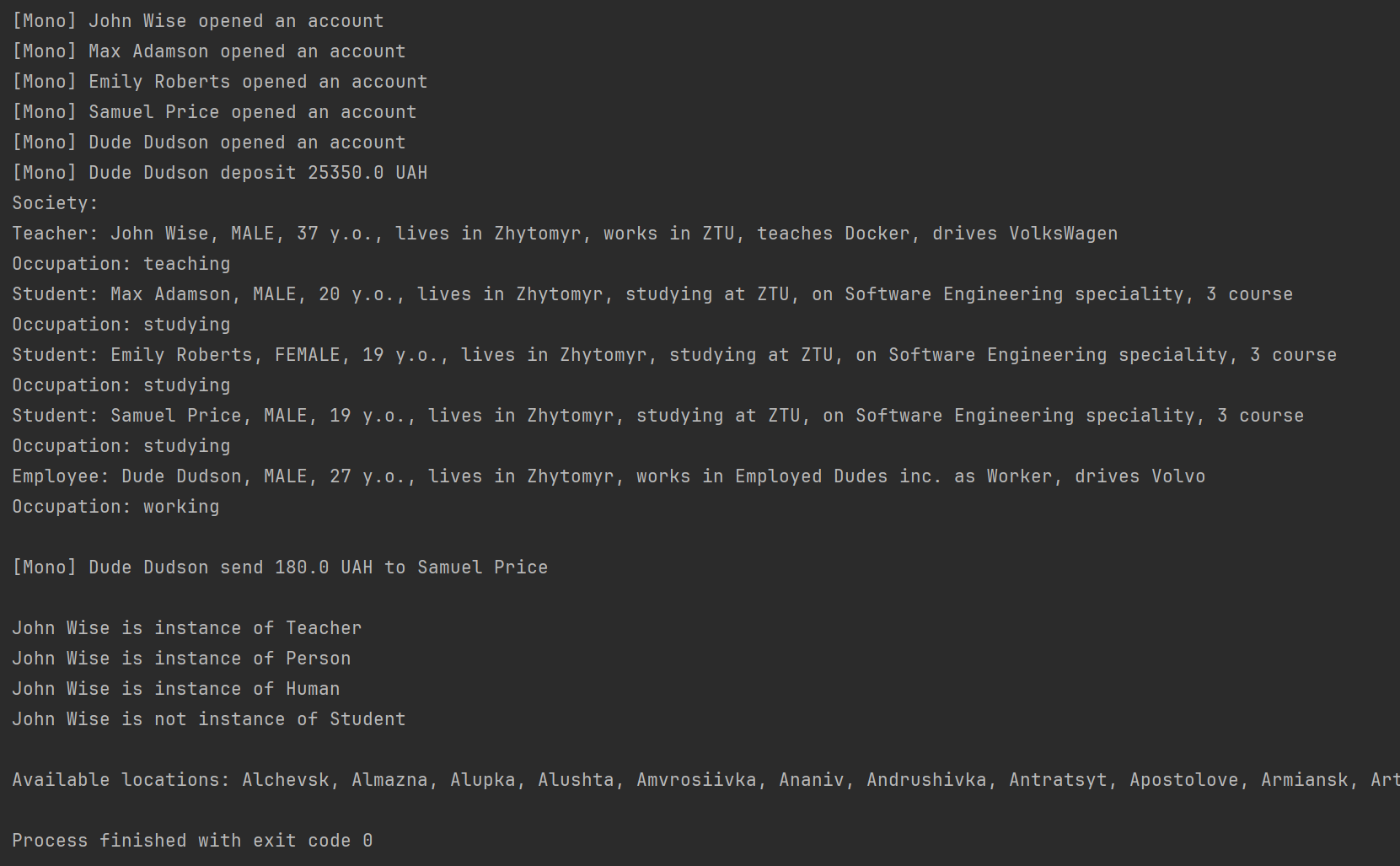
}

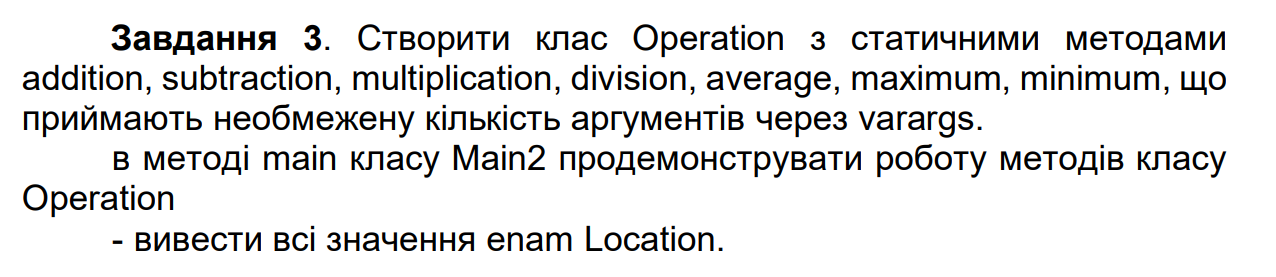
public static boolean isInstanceOf(Object instance, Class<?> of){

return of.isInstance(instance);

}

}





public class Operation {

public static double addition(double... values){

double result = 0;

for(double value: values){

result += value;

}

return result;

}

public static double subtraction(double... values){

int i = 0;

double result = values[i];

for(++i; i < values.length; i += 1){

result -= values[i];

}

return result;

}

public static double multiplication(double... values){

double result = 1;

for(double value: values){

result \*= value;

}

return result;

}

public static double division(double... values){

int i = 0;

double result = values[i];

for(++i; i < values.length; i += 1){

result /= values[i];

}

return result;

}

public static double average(double... values){

return addition(values) / values.length;

}

public static double maximum(double... values){

double result = values[0];

for(double value: values){

if(value > result){

result = value;

}

}

return result;

}

public static double minimum(double... values){

double result = values[0];

for(double value: values){

if(value < result){

result = value;

}

}

return result;

}

}

public class Main {

public static void main(String[] args) {

double[] values = new double[]{78, 52, 7, 59, 10, 82, 16, 70, 87, 40};

System.out.println("Addition: " + Operation.addition(values));

System.out.println("Subtraction: " + Operation.subtraction(values));

System.out.println("Multiplication: " + Operation.multiplication(values));

System.out.println("Division: " + Operation.division(values));

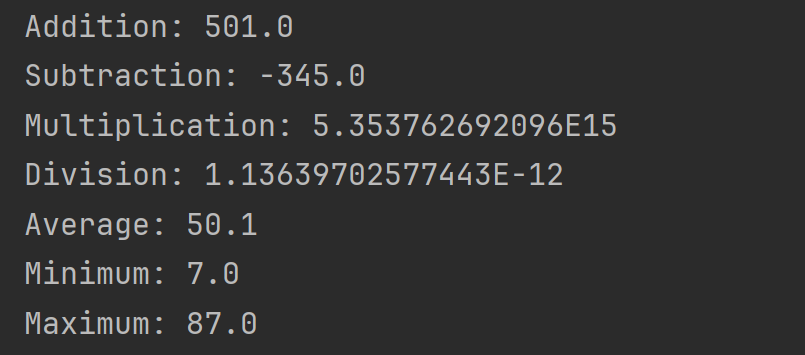
System.out.println("Average: " + Operation.average(values));

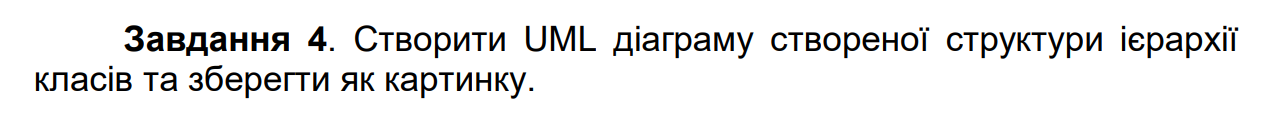
System.out.println("Minimum: " + Operation.minimum(values));

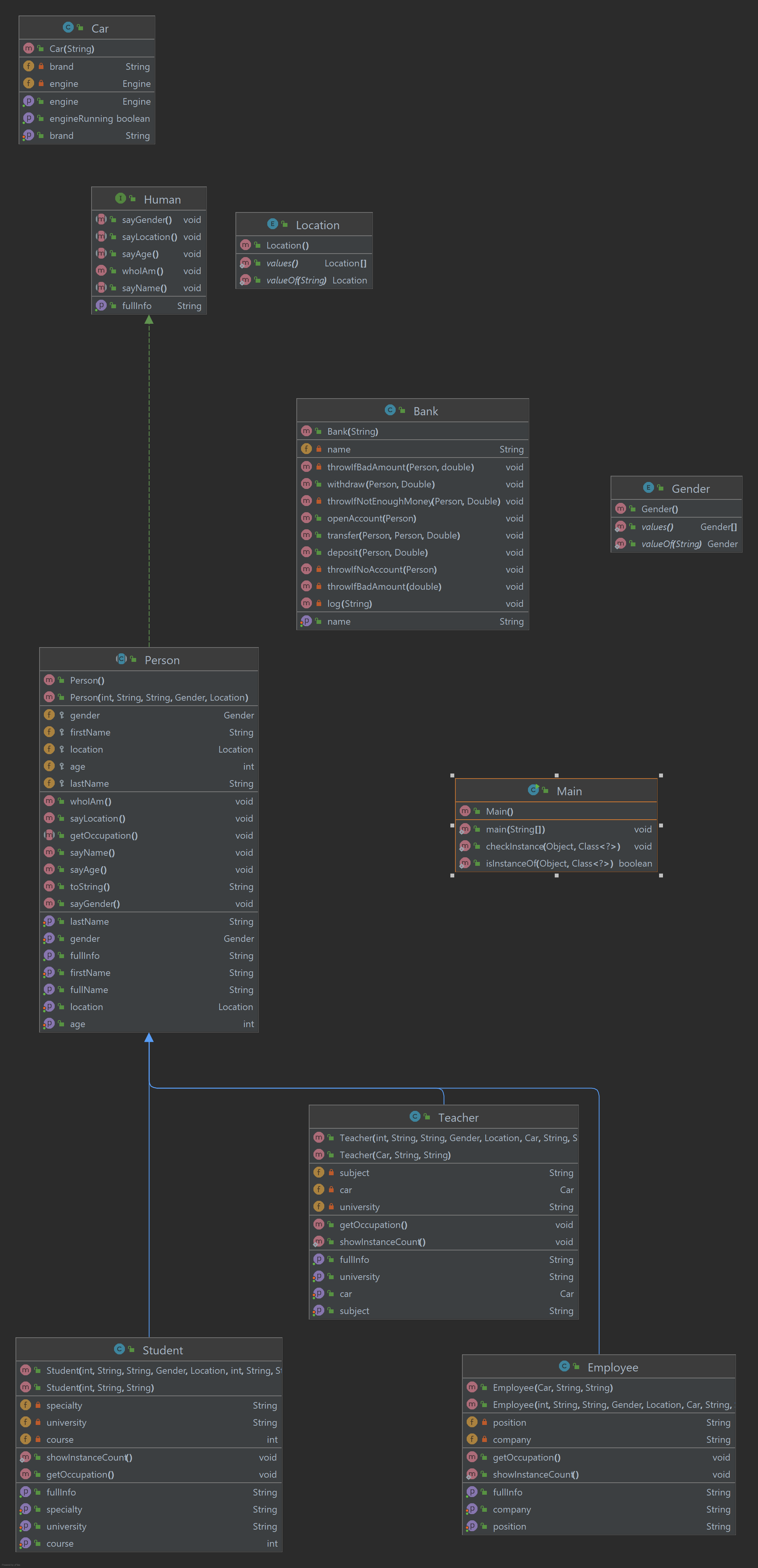
System.out.println("Maximum: " + Operation.maximum(values));

}

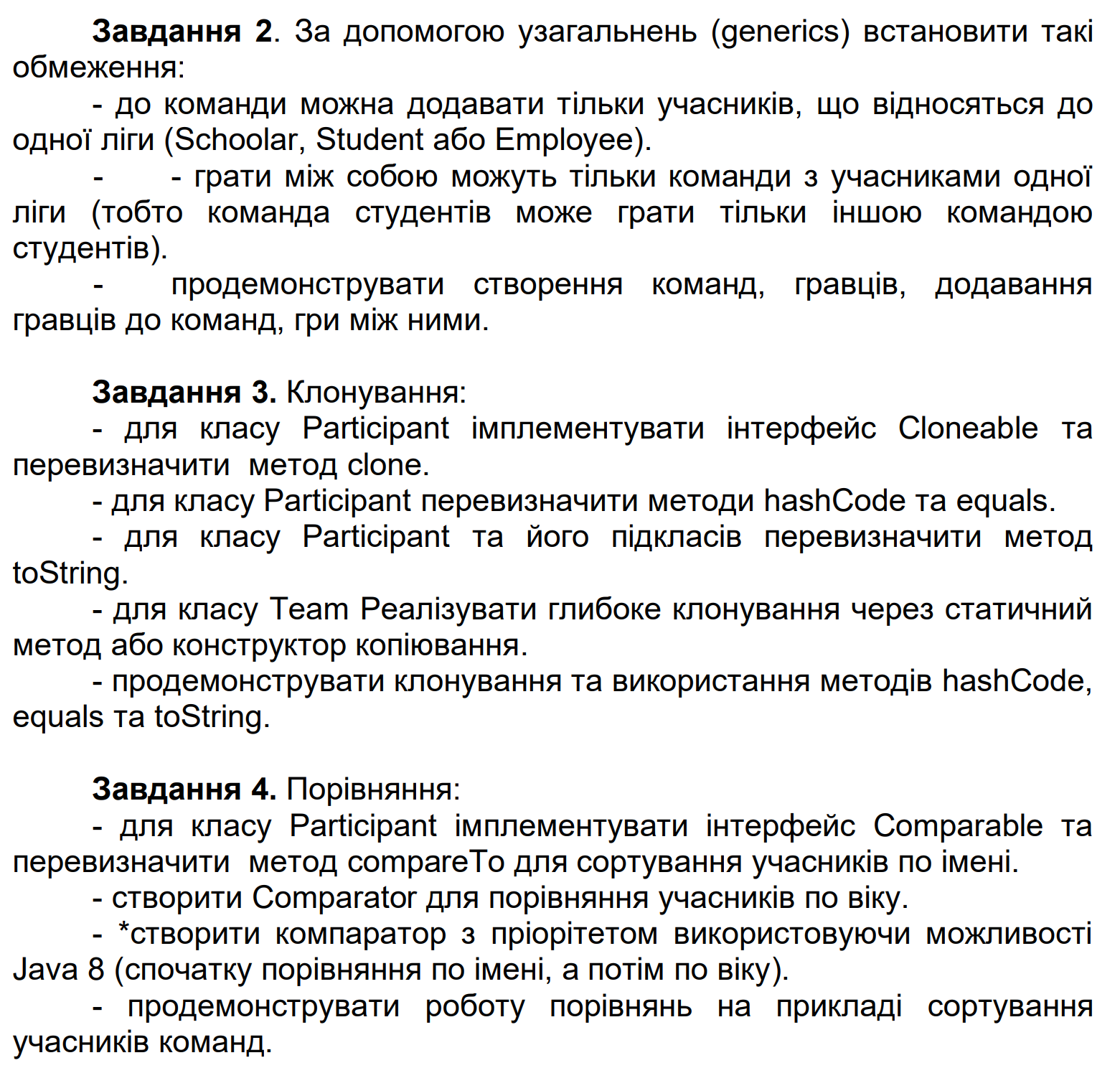
}







**Лабораторна робота №3**



public class Employee extends Participant{

public Employee(String name, int age) {

super(name, age);

}

@Override

public String toString() {

return "Employee [name=" + getName() + ", age=" + getAge() + "]";

}

}

public abstract class Participant implements Cloneable, Comparable<Participant>{

public static Comparator<Participant> ageComparator = new Comparator<>() {

@Override

public int compare(Participant p1, Participant p2) {

return p1.getAge() - p2.getAge();

}

};

private String name;

private int age;

public Participant(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public void setName(String name) {

this.name = name;

}

public void setAge(int age) {

this.age = age;

}

@Override

public Participant clone() {

try {

Participant clone = (Participant) super.clone();

return clone;

} catch (CloneNotSupportedException e) {

throw new AssertionError();

}

}

@Override

public String toString() {

return "Participant [name=" + name + ", age=" + age + "]";

}

// @Override

// public int hashCode() {

public int customHashCode() {

return name.hashCode() \* getAge();

}

public boolean equals(Participant participant) {

return participant.toString().equals(toString());

}

@Override

public int compareTo(Participant participant) {

return name.compareTo(participant.getName());

}

}

public class Schoolar extends Participant{

public Schoolar(String name, int age) {

super(name, age);

}

@Override

public String toString() {

return "Schoolar [name=" + getName() + ", age=" + getAge() + "]";

}

}

public class Student extends Participant{

public Student(String name, int age) {

super(name, age);

}

@Override

public String toString() {

return "Student [name=" + getName() + ", age=" + getAge() + "]";

}

}

public class Team<T extends Participant> {

private String name;

private List<T> participants = new ArrayList<>();

public Team(String name) {

this.name = name;

}

public Team(Team<T> team) {

this.name = team.name;

this.participants = new ArrayList<>();

for (T participant : team.participants) {

@SuppressWarnings("unchecked")

T clonedParticipant = (T) participant.clone();

participants.add(clonedParticipant);

}

}

public void addNewParticipant(T participant) {

participants.add(participant);

if(participant != null) {

System.out.println("To the team " + name + " was added participant " + ((Participant) participant).getName());

} else {

System.out.println("To the team " + name + " was added participant " + participant);

}

}

public void playWith(Team<T> team) {

String winnerName;

Random random = new Random();

int i = random.nextInt(2);

if(i == 0) {

winnerName = this.name;

} else {

winnerName = team.name;

}

System.out.println("The team " + winnerName + " is winner!");

}

public String getName() {

return name;

}

public List<T> getParticipants() {

return participants;

}

public void setName(String name) {

this.name = name;

}

public void setParticipants(List<T> participants) {

this.participants = participants;

}

}

public class Main {

public static void main(String[] args) {

Student student1 = new Student("Student 1", 20);

Student student2 = new Student("Student 2", 18);

Student student3 = new Student("Student 3", 19);

Student student4 = new Student("Student 4", 19);

Team<Student> studentTeam1 = new Team<>("Student team 1");

Team<Student> studentTeam2 = new Team<>("Student team 2");

Schoolar schoolar1 = new Schoolar("Schoolar 1", 15);

Schoolar schoolar2 = new Schoolar("Schoolar 2", 16);

Schoolar schoolar3 = new Schoolar("Schoolar 3", 14);

Schoolar schoolar4 = new Schoolar("Schoolar 4", 15);

Team<Schoolar> schoolarTeam1 = new Team<>("Schoolar team 1");

Team<Schoolar> schoolarTeam2 = new Team<>("Schoolar team 2");

System.out.println("==========TASK 1==========");

studentTeam1.addNewParticipant(student1);

studentTeam1.addNewParticipant(student2);

studentTeam2.addNewParticipant(student3);

studentTeam2.addNewParticipant(student4);

schoolarTeam1.addNewParticipant(schoolar1);

schoolarTeam1.addNewParticipant(schoolar2);

schoolarTeam2.addNewParticipant(schoolar3);

schoolarTeam2.addNewParticipant(schoolar4);

// studentTeam1.addNewParticipant(schoolar1); - Will show type error

studentTeam1.playWith(studentTeam2);

// studentTeam1.playWith(schoolarTeam1); - Will show type error

System.out.println();

System.out.println("==========TASK 2==========");

Student student1Clone = (Student)student1.clone();

System.out.println(student1);

System.out.println(student1Clone);

System.out.println(student1.hashCode());

System.out.println(student1Clone.hashCode());

System.out.println(student1.customHashCode());

System.out.println(student1Clone.customHashCode());

System.out.println(student1.equals(student1Clone));

System.out.println();

Team<Student> studentTeam1Clone = new Team<>(studentTeam1);

System.out.println(studentTeam1);

System.out.println(studentTeam1Clone);

System.out.println(studentTeam1.getName());

System.out.println(studentTeam1Clone.getName());

System.out.println(studentTeam1.getParticipants());

System.out.println(studentTeam1Clone.getParticipants());

System.out.println(studentTeam1.getParticipants().getFirst().hashCode());

System.out.println(studentTeam1Clone.getParticipants().getFirst().hashCode());

System.out.println(studentTeam1.getParticipants().getFirst().customHashCode());

System.out.println(studentTeam1Clone.getParticipants().getFirst().customHashCode());

studentTeam1Clone.getParticipants().getFirst().setName("Cloned with Team!");

System.out.println(studentTeam1.getParticipants());

System.out.println(studentTeam1Clone.getParticipants());

System.out.println(studentTeam1.getParticipants().getFirst().customHashCode());

System.out.println(studentTeam1Clone.getParticipants().getFirst().customHashCode());

System.out.println();

System.out.println("==========TASK 3==========");

studentTeam1.addNewParticipant(student4);

studentTeam1.addNewParticipant(student3);

System.out.println(studentTeam1.getParticipants());

Collections.sort(studentTeam1.getParticipants());

System.out.println(studentTeam1.getParticipants());

studentTeam1.getParticipants().sort(Participant.ageComparator);

System.out.println(studentTeam1.getParticipants());

studentTeam1.getParticipants().sort(Comparator.comparing(Participant::getName).thenComparing(Participant::getAge));

System.out.println(studentTeam1.getParticipants());

}

}

