**Міністерство освіти і науки України**

**Національний технічний університет**

**«Дніпровська політехніка»**

****

**ЗВІТ**

**Практична робота №4**

**з дисципліни**

# «Програмування в середовищі Java»

**Виконав:**

студент гр. 122-21-3

Кабаченко О. В.

**Прийняв:**

Доцент каф. САУ

Алексєєв О. М.

**м. Дніпро**

**2025  рік**

**Хід роботи**

1) Для початку в існуючому проекті було створено новий пакет «lab4», до якого додано пакети– «modelс, «view», «controller», а також файл «Run». На рисунку 1 наведено результат роботи.

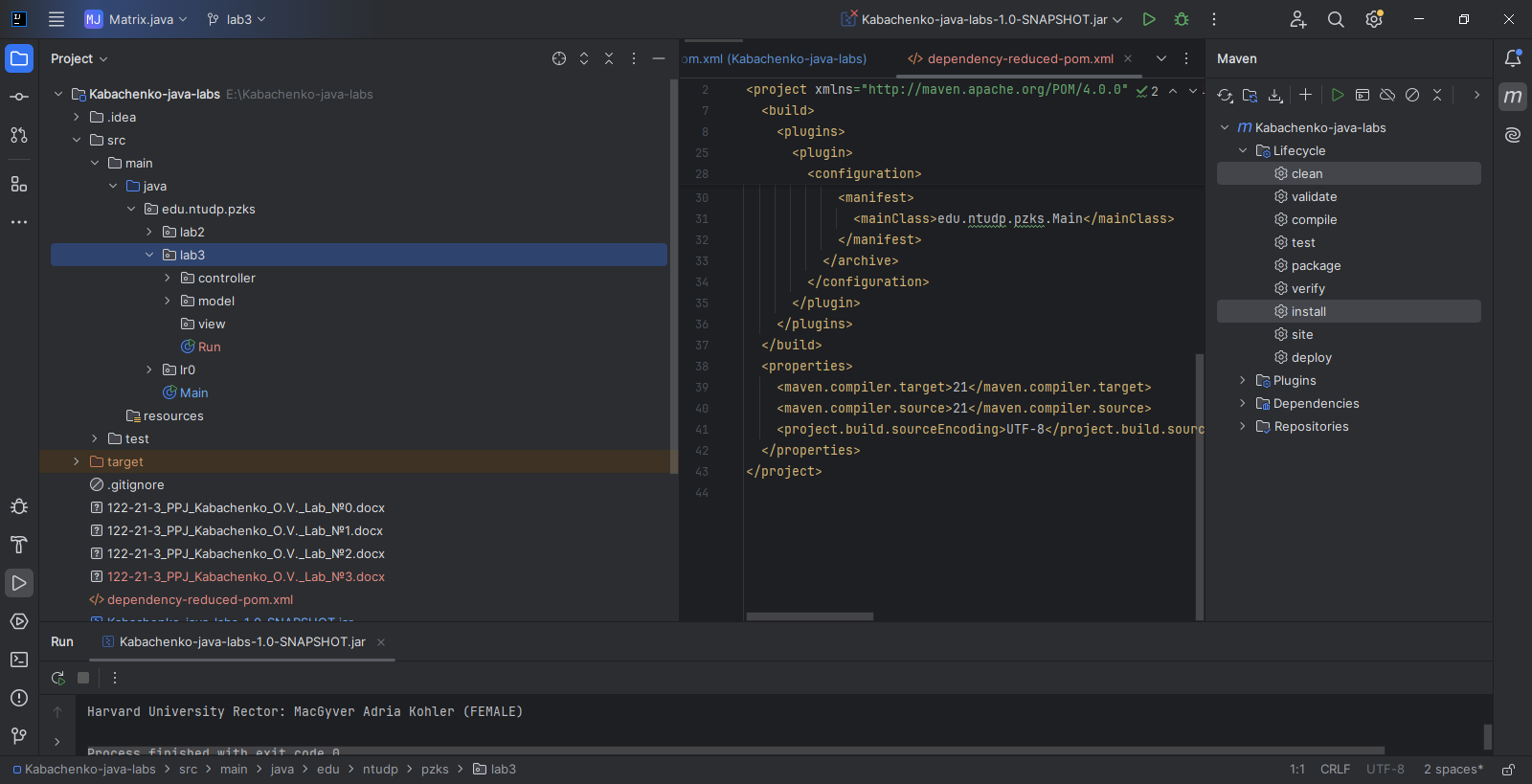


Рисунок 1 – Базова структура нового пакету

2) Далі в пакеті моделей було створено 8 файлів – «CreatureInterface», «Department», «Faculty», «Group», «Human», «OrganizationalUnit», «Student» та «University». «OrganizationalUnit» являє собою абстрактний клас, від якого унаслідуються «Department», «Faculty», «Group» та «University». «CreatureInterface» являє собою інтерфейс, який реалізує клас «Human». Від «Human» унаслідуються клас «Student». Нижче наведено програмний код усіх файлів.

Програмний код класу CreatureInterface:

package edu.ntudp.pzks.lab4.model;  
  
public interface CreatureInterface {  
 public static enum Sex {  
 *MALE*, *FEMALE* }  
 public Sex getGender();  
 public String getName();  
}

Програмний код класу Department:

package edu.ntudp.pzks.lab4.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Department extends OrganizationalUnit {  
 private List<Group> groups = new ArrayList<>();  
  
 public Department(String name, Human head) {  
 super(name, head);  
 }  
  
 public Department(String name, Human head, List<Group> groups) {  
 super(name, head);  
 this.groups = groups;  
 }  
  
 public void addGroup(Group group) {  
 groups.add(group);  
 }  
  
 public List<Group> getGroups() {  
 return groups;  
 }  
  
 public void setGroups(List<Group> groups) {  
 this.groups = groups;  
 }  
  
 @Override  
 public String toString() {  
 return name +" Head of department: " + head;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 if (!super.equals(o)) return false;  
 Department that = (Department) o;  
 return Objects.*equals*(groups, that.groups);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, head, groups);  
 }  
}

Програмний код класу Faculty:

package edu.ntudp.pzks.lab4.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Faculty extends OrganizationalUnit {  
 private List<Department> departments = new ArrayList<>();  
  
 public Faculty(String name, Human head) {  
 super(name, head);  
 }  
  
 public Faculty(String name, Human head, List<Department> departments) {  
 super(name, head);  
 this.departments = departments;  
 }  
  
 public void addDepartment(Department department) {  
 departments.add(department);  
 }  
  
 public List<Department> getDepartments() {  
 return departments;  
 }  
  
 public void setDepartments(List<Department> departments) {  
 this.departments = departments;  
 }  
  
 @Override  
 public String toString() {  
 return name +" Dean: " + head;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 if (!super.equals(o)) return false;  
 Faculty that = (Faculty) o;  
 return Objects.*equals*(departments, that.departments);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, head, departments);  
 }  
}

Програмний код класу Group:

package edu.ntudp.pzks.lab4.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Group extends OrganizationalUnit {  
 private List<Student> students = new ArrayList<>();  
  
 public Group(String name, Human head) {  
 super(name, head);  
 }  
  
 public Group(String name, Human head, List<Student> students) {  
 super(name, head);  
 this.students = students;  
 }  
  
 public void addStudent(Student student) {  
 students.add(student);  
 }  
  
 public List<Student> getStudents() {  
 return students;  
 }  
  
 public void setStudents(List<Student> students) {  
 this.students = students;  
 }  
  
 @Override  
 public String toString() {  
 return name + " Curator: " + head;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 if (!super.equals(o)) return false;  
 Group that = (Group) o;  
 return Objects.*equals*(students, that.students);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, head, students);  
 }  
  
}

Програмний код класу Human:

package edu.ntudp.pzks.lab4.model;  
import java.util.UUID;  
import java.util.Objects;  
  
public class Human implements CreatureInterface{  
 protected String firstName;  
 protected String middleName;  
 protected String lastName;  
 protected Sex gender;  
 protected final UUID id;  
  
 public Human(String firstName, String middleName, String lastName, Sex gender) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.middleName = middleName;  
 this.gender = gender;  
 this.id = UUID.*randomUUID*();  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public String getMiddleName() {  
 return middleName;  
 }  
  
 public Sex getGender() {  
 return gender;  
 }  
  
 public String getName() {  
 return middleName + " " + firstName + " " + lastName;  
 }  
  
 public UUID getId() {  
 return id;  
 }  
  
 @Override  
 public String toString() {  
 return middleName + " " + firstName + " " + lastName + " (" + gender + ")";  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 Human human = (Human) o;  
 return Objects.*equals*(firstName, human.firstName) && Objects.*equals*(lastName, human.lastName) && Objects.*equals*(middleName, human.middleName) && Objects.*equals*(id, human.id);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(firstName, lastName, middleName, id);  
 }  
}

Програмний код класу OrganizationalUnit:

package edu.ntudp.pzks.lab4.model;  
  
import java.util.Objects;  
  
abstract public class OrganizationalUnit {  
 protected String name;  
 protected Human head;  
  
 public OrganizationalUnit(String name, Human head) {  
 this.name = name;  
 this.head = head;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public Human getHead() {  
 return head;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setHead(Human head) {  
 this.head = head;  
 }  
  
 @Override  
 public String toString() {  
 return name + " Head: " + head;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 OrganizationalUnit that = (OrganizationalUnit) o;  
 return Objects.*equals*(name, that.name) && Objects.*equals*(head, that.head);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, head);  
 }  
}

Програмний код класу Student:

package edu.ntudp.pzks.lab4.model;  
  
import java.util.Objects;  
  
public class Student extends Human {  
 private final String recordBookID;  
  
 public Student(String firstName, String middleName, String lastName, Sex gender, String recordBookID) {  
 super(firstName, lastName, middleName, gender);  
 this.recordBookID = recordBookID;  
 }  
  
 public String getStudentID() {  
 return recordBookID;  
 }  
  
 public String getRecordBookID() {  
 return recordBookID;  
 }  
  
 @Override  
 public String toString() {  
 return super.toString() + " - Student, record book id: " + recordBookID;  
 }  
  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 if (!super.equals(o)) return false;  
 Student that = (Student) o;  
 return Objects.*equals*(recordBookID, that.recordBookID);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(super.hashCode(), recordBookID);  
 }  
}

Програмний код класу University:

package edu.ntudp.pzks.lab4.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class University extends OrganizationalUnit {  
 private List<Faculty> faculties = new ArrayList<>();  
  
 public University(String name, Human head) {  
 super(name, head);  
 }  
  
 public University(String name, Human head, List<Faculty> faculties) {  
 super(name, head);  
 this.faculties = faculties;  
 }  
  
 public void addFaculty(Faculty faculty) {  
 faculties.add(faculty);  
 }  
  
 public List<Faculty> getFaculties() {  
 return faculties;  
 }  
  
 public void setFaculties(List<Faculty> faculties) {  
 this.faculties = faculties;  
 }  
  
 @Override  
 public String toString() {  
 return name + " Rector: " + head;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 if (!super.equals(o)) return false;  
 University that = (University) o;  
 return Objects.*equals*(faculties, that.faculties);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, head, faculties);  
 }  
}

3) На цьому кроці в пакеті контролерів було створено наступні файли: «DepartmentCreator», «FacultyCreator», «GroupCreator», «HumanCreator», «PersonCreator», «StudentCreator», «UniversityCreator». «PersonCreator» є абстрактнім класом, від якого унаслідуються такі класи як «HumanCreator» та «StudentCreator». Нижче наведено програмний код усіх файлів.

Програмний код класу DepartmentCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import edu.ntudp.pzks.lab4.model.Department;  
import edu.ntudp.pzks.lab4.model.Group;  
import edu.ntudp.pzks.lab4.model.Human;  
  
import java.util.\*;  
  
public class DepartmentCreator {  
 private static final Random *random* = new Random();  
  
 private static final Map<String, List<String>> *departments* = Map.*of*(  
 "Department of Software Engineering of Computer Systems", Arrays.*asList*("122", "121"),  
 "Department of Information Technology and Computer Engineering", Arrays.*asList*("123","126"),  
 "Department of Applied Economics, Entrepreneurship, and Public Administration", Arrays.*asList*("051", "076", "075"),  
 "Department of Management", Arrays.*asList*("073", "072"),  
 "Department of Chemistry and Chemical Engineering", Arrays.*asList*("161", "102"),  
 "Department of Oil and Gas Engineering and Drilling", Arrays.*asList*("185", "015.35", "184"),  
 "Department of Systems Analysis and Management", Arrays.*asList*("124", "125")  
 );  
  
 private static Map.Entry<String, List<String>> getRandomDepartmentName() {  
 List<Map.Entry<String, List<String>>> entries = new ArrayList<>(*departments*.entrySet());  
 return entries.get(*random*.nextInt(entries.size()));  
 }  
  
 public static Map.Entry<String, List<String>> getDepartmentDataByName(String departmentName) {  
 if (*departments*.containsKey(departmentName)) {  
 return Map.*entry*(departmentName, *departments*.get(departmentName));  
 } else {  
 return null;  
 }  
 }  
  
 public static Department createTypicalDepartment(int groupCnt, boolean iscascadeSubdivisions) {  
 return *createTypicalDepartment*(*getRandomDepartmentName*(), groupCnt, iscascadeSubdivisions);  
 }  
  
 public static Department createTypicalDepartment() {  
 return *createTypicalDepartment*(*getRandomDepartmentName*());  
 }  
  
 public static Department createTypicalDepartment(Map.Entry<String, List<String>> departmentData) {  
 return *createTypicalDepartment*(departmentData, *random*.nextInt(3)+1);  
 }  
  
 public static Department createTypicalDepartment(Map.Entry<String, List<String>> departmentData, int specialtiesCount) {  
 return *createTypicalDepartment*(departmentData, specialtiesCount, false);  
 }  
  
 public static Department createTypicalDepartment(Map.Entry<String, List<String>> departmentData, int specialtiesCount, boolean iscascadeSubdivisions) {  
 String departmentName = departmentData.getKey();  
 List<String> specialties = departmentData.getValue();  
  
 List<Group> groups = new ArrayList<>();  
 Human head = HumanCreator.*createTypicalHuman*();  
  
 specialtiesCount = Math.*min*(specialties.size(), specialtiesCount);  
 List<String> selectedSpecialties = new ArrayList<>();  
  
 Collections.*shuffle*(specialties);  
 for (int i = 0; i < specialtiesCount; i++) {  
 selectedSpecialties.add(specialties.get(i));  
 }  
  
 int numGroups = 1;  
 for (String specialty : selectedSpecialties) {  
 if (!iscascadeSubdivisions){  
 numGroups = *random*.nextInt(3) + 1;  
 }  
  
 int year = *random*.nextInt(6) + 20;  
 for (int i = 0; i < numGroups; i++) {  
 String groupName = (specialty +"-"+year+"-"+(i+1));  
  
 Group group = GroupCreator.*createTypicalGroup*(groupName);  
  
 if (iscascadeSubdivisions){  
 group = GroupCreator.*createTypicalGroup*(groupName, specialtiesCount);  
 }  
 groups.add(group);  
 }  
 }  
 return new Department(departmentName, head, groups);  
 }  
  
 public static Department createDepartment(String name, Human head, List<Group> groups) {  
 return new Department(name, head, groups);  
 }  
  
 public static Department createEmptyDepartment(String name, Human head) {  
 return new Department(name, head);  
 }  
}

Програмний код класу FacultyCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import edu.ntudp.pzks.lab4.model.Department;  
import edu.ntudp.pzks.lab4.model.Faculty;  
import edu.ntudp.pzks.lab4.model.Human;  
  
import java.util.\*;  
  
public class FacultyCreator {  
 private static final Random *random* = new Random();  
  
 public static final Map<String, List<String>> *faculties* = Map.*of*(  
 "Faculty of Information Technologies", Arrays.*asList*("Department of Software Engineering of Computer Systems", "Department of Information Technology and Computer Engineering", "Department of Systems Analysis and Management"),  
 "Faculty of Management", Arrays.*asList*("Department of Management", "Department of Applied Economics, Entrepreneurship, and Public Administration"),  
 "Faculty of Natural Sciences and Technologies", Arrays.*asList*("Department of Oil and Gas Engineering and Drilling", "Department of Chemistry and Chemical Engineering")  
 );  
  
 private static Map.Entry<String, List<String>> getRandomFacultyName() {  
 List<Map.Entry<String, List<String>>> entries = new ArrayList<>(*faculties*.entrySet());  
 return entries.get(*random*.nextInt(entries.size()));  
 }  
  
 public static Faculty createTypicalFaculty() {  
 return *createTypicalFaculty*(*getRandomFacultyName*());  
 }  
  
 public static Faculty createTypicalFaculty(Map.Entry<String, List<String>> facultyData) {  
 return *createTypicalFaculty*(facultyData, *random*.nextInt(3)+1, false);  
 }  
  
 public static Faculty createTypicalFaculty(int departmentsCount, boolean iscascadeSubdivisions) {  
 return *createTypicalFaculty*(*getRandomFacultyName*(), departmentsCount, iscascadeSubdivisions);  
 }  
  
 public static Faculty createTypicalFaculty(Map.Entry<String, List<String>> facultyData, int departmentsCount, boolean iscascadeSubdivisions) {  
 String facultyName = facultyData.getKey();  
 List<String> departments = facultyData.getValue();  
 Human head = HumanCreator.*createTypicalHuman*();  
 Faculty faculty = new Faculty(facultyName, head);  
  
 departmentsCount = Math.*min*(departments.size(), departmentsCount);  
  
 Collections.*shuffle*(departments);  
 for (int i = 0; i < departmentsCount; i++) {  
 String departmentName = departments.get(i);  
  
 Map.Entry<String, List<String>> departmentData = DepartmentCreator.*getDepartmentDataByName*(departmentName);  
 if (departmentData != null) {  
 Department department = DepartmentCreator.*createTypicalDepartment*(departmentData);  
  
 if (iscascadeSubdivisions) {  
 department = DepartmentCreator.*createTypicalDepartment*(departmentData, departmentsCount, iscascadeSubdivisions);  
 }  
 faculty.addDepartment(department);  
 }  
 }  
 return faculty;  
 }  
  
 public static Faculty createFaculty(String name, Human head, List<Department> departments) {  
 return new Faculty(name, head, departments);  
 }  
  
 public static Faculty createEmptyFaculty(String name, Human head) {  
 return new Faculty(name, head);  
 }  
}

Програмний код класу GroupCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import edu.ntudp.pzks.lab4.model.Group;  
import edu.ntudp.pzks.lab4.model.Human;  
import edu.ntudp.pzks.lab4.model.Student;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class GroupCreator{  
 private static final Random *random* = new Random();  
  
 private static String generateGroupName() {  
 int facultyNumber = *random*.nextInt(293)+1;  
 int admissionYear = *random*.nextInt(6) + 20;  
 int groupNumber = *random*.nextInt(4)+1;  
  
 return String.*format*("%d-%d-%d", facultyNumber, admissionYear, groupNumber);  
 }  
  
 public static Group createTypicalGroup() {  
 return *createTypicalGroup*(*generateGroupName*());  
 }  
  
 public static Group createTypicalGroup(String groupName) {  
 return *createTypicalGroup*(groupName, *random*.nextInt(11) + 20);  
 }  
  
 public static Group createTypicalGroup(int studentsInGroup) {  
 return *createTypicalGroup*(*generateGroupName*(), studentsInGroup);  
 }  
  
 public static Group createTypicalGroup(String groupName, int studentCount){  
 List<Student> students = new ArrayList<>();  
  
 for (int i = 0; i < studentCount; i++) {  
 students.add(StudentCreator.*createTypicalStudent*());  
 }  
  
 Human head = HumanCreator.*createTypicalHuman*();  
  
 return new Group(groupName, head, students);  
 }  
  
 public static Group createGroup(String name, Human head, List<Student> students) {  
 return new Group(name, head, students);  
 }  
  
 public static Group createEmptyGroup(String name, Human head) {  
 return new Group(name, head);  
 }  
}

Програмний код класу HumanCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import edu.ntudp.pzks.lab4.model.Human;  
  
public class HumanCreator extends PersonCreator {  
 public static Human createTypicalHuman() {  
 String[] names = *generateRandomFullName*();  
 Human.Sex gender = *getRandomGender*();  
 return new Human(names[0], names[1], names[2], gender);  
 }  
  
 public static Human createHuman(String firstName, String middleName, String lastName, Human.Sex gender) {  
 return new Human (firstName, middleName, lastName, gender);  
 }  
}

Програмний код класу PersonCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import com.github.javafaker.Faker;  
import edu.ntudp.pzks.lab4.model.Human;  
import java.util.Random;  
  
abstract public class PersonCreator {  
 public static final Random *random* = new Random();  
 public static final Faker *faker* = new Faker();  
  
 protected static Human.Sex getRandomGender() {  
 return Human.Sex.*values*()[*random*.nextInt(Human.Sex.*values*().length)];  
 }  
  
 protected static String[] generateRandomFullName() {  
 return new String[]{  
 *faker*.name().firstName(),  
 *faker*.name().lastName(),  
 *faker*.name().lastName()  
 };  
 }  
}

Програмний код класу StudentCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import edu.ntudp.pzks.lab4.model.Student;  
  
public class StudentCreator extends PersonCreator {  
 public static Student createTypicalStudent() {  
 String[] names = *generateRandomFullName*();  
 Student.Sex gender = *getRandomGender*();  
 String recordBookID = String.*format*("%06d", *random*.nextInt(1000000));  
  
 return new Student (names[0], names[1], names[2], gender,recordBookID);  
 }  
  
 public static Student createStudent(String firstName, String middleName, String lastName, Student.Sex gender, String recordBookID) {  
 return new Student (firstName, middleName, lastName, gender, recordBookID);  
 }  
}

Програмний код класу UniversityCreator:

package edu.ntudp.pzks.lab4.controller;  
  
import edu.ntudp.pzks.lab4.model.Faculty;  
import edu.ntudp.pzks.lab4.model.University;  
import edu.ntudp.pzks.lab4.model.Human;  
  
import java.util.\*;  
  
import static edu.ntudp.pzks.lab3.controller.PersonCreator.*random*;  
  
public class UniversityCreator {  
 private static final List<String> *universities* = Arrays.*asList*(  
 "Harvard University",  
 "Stanford University",  
 "Dnipro University of Technology",  
 "Igor Sikorsky Kyiv Polytechnic Institute",  
 "Lviv Polytechnic National University"  
 );  
  
 private static String getRandomUniversityName() {  
 int index = *random*.nextInt(*universities*.size());  
 return *universities*.get(index);  
 }  
  
 public static University createTypicalUniversity() {  
 return *createTypicalUniversity*(*getRandomUniversityName*() );  
 }  
  
 public static University createTypicalUniversity(String universityName) {  
 return *createTypicalUniversity*(universityName, *random*.nextInt(3)+1, false);  
 }  
  
  
 public static University createTypicalUniversity(int facultiesCount, boolean iscascadeSubdivisions) {  
 return *createTypicalUniversity*(*getRandomUniversityName*(), facultiesCount, iscascadeSubdivisions);  
 }  
  
  
 public static University createTypicalUniversity(String universityName, int facultiesCount, boolean iscascadeSubdivisions) {  
 Human head = HumanCreator.*createTypicalHuman*();  
 University university = new University(universityName, head);  
 List<Map.Entry<String, List<String>>> facultyList = new ArrayList<>(FacultyCreator.*faculties*.entrySet());  
  
 facultiesCount = Math.*min*(FacultyCreator.*faculties*.size(), facultiesCount);  
  
 Collections.*shuffle*(facultyList);  
 for (int i = 0; i < facultiesCount; i++) {  
 Faculty faculty = FacultyCreator.*createTypicalFaculty*(facultyList.get(i));  
  
 if(iscascadeSubdivisions){  
 faculty = FacultyCreator.*createTypicalFaculty*(facultyList.get(i), facultiesCount, iscascadeSubdivisions);  
 }  
 university.addFaculty(faculty);  
 }  
 return university;  
 }  
  
 public static University createUniversity(String name, Human head, List<Faculty> faculties) {  
 return new University(name, head, faculties);  
 }  
  
 public static University createEmptyUniversity(String name, Human head) {  
 return new University(name, head);  
 }  
}

4) Після попереднього кроку у файлі Run було написано код для створення університету(його наведено нижче). На рисунку 2 наведено результат виконання програми.

package edu.ntudp.pzks.lab4;  
  
import edu.ntudp.pzks.lab4.model.University;  
  
import static edu.ntudp.pzks.lab4.controller.UniversityCreator.*createTypicalUniversity*;  
  
public class Run {  
 public static void main(String[] args){  
 University university = *createTypicalUniversity*();  
 System.*out*.println(university);  
 }

}

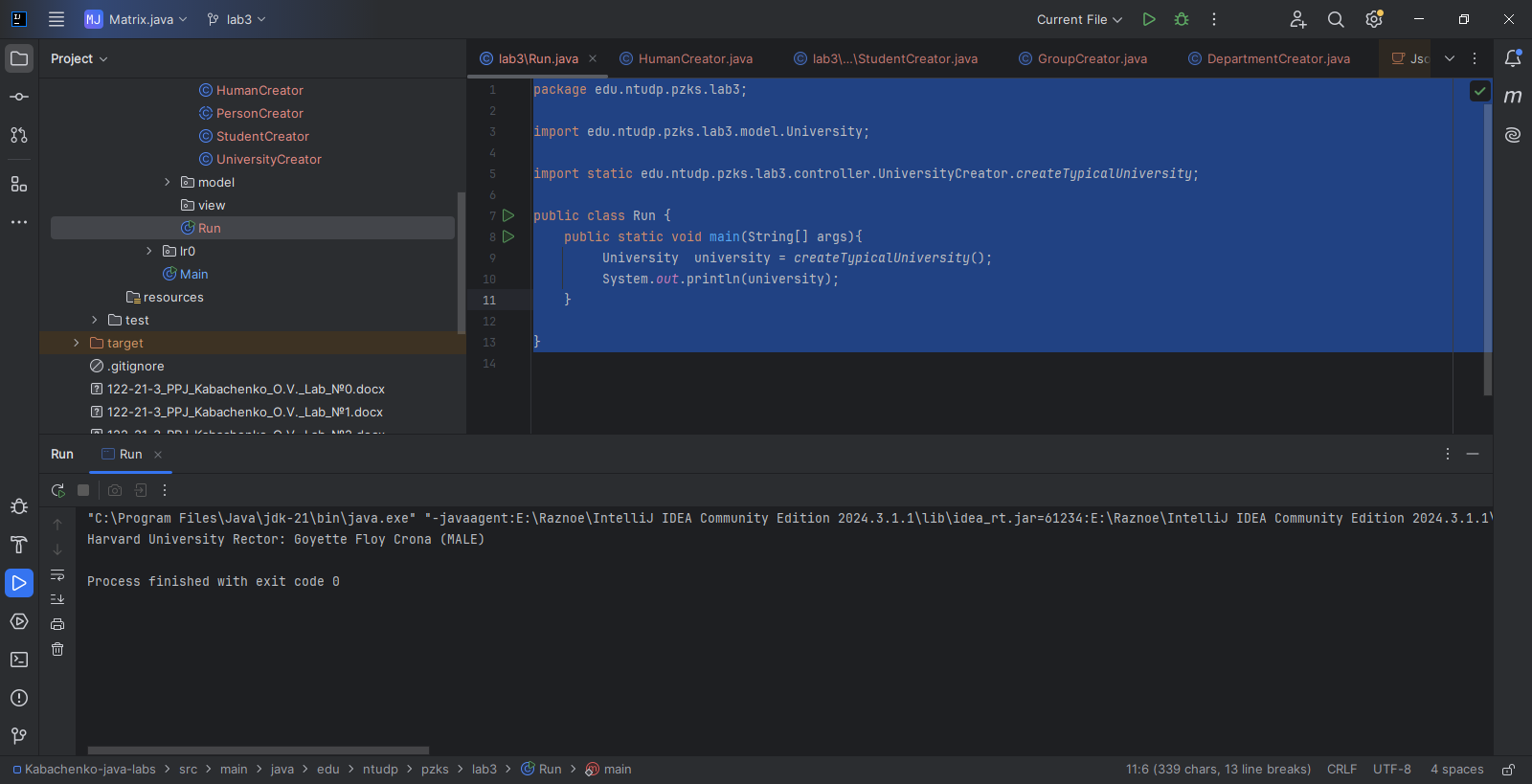


Рисунок 2 – Результат створення університету.

**Висновки**

У ході виконання лабораторної роботи розглянуто принципи ООП, а також досліджено використання абстрактних класів та інтерфейсів.