Міністерство освіти і науки України

Київський політехнічний інститут ім. Ігоря Сікорського

Теплоенергетичний факультет

Кафедра АПЕПС

ЗВІТ

про виконання лабораторної роботи №4

з дисципліни «Технології конструювання програмного забезпечення»

**«Использование коллекций в Javа»**

**Виконала:**

Студентка 2 курсу, групи ТІ-01

Круть Катерина Олександрівна

Дата: 22.11.21

**Перевірив:**

доцент, к.ф.-м.н.

Тарнавський Ю.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Варіант №14

Завдання: 3,5,19,29

**Завдання №9:**

Создайте приложение для просмотра списка элементов – целых чисел (типа Integer) и изменения элементов списка в заданном диапазоне индексов. Список (10 чисел) является объектом класса  ArrayList. Предусмотреть возможность расчета количества повторов элементов, для чего использовать HashSet.

**Код програми:**

package task1;  
  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;  
import java.util.Scanner;  
  
public class Task\_1 {  
 public static void main(String[] args) {  
 List<Integer> numbers = new ArrayList<>();  
 Scanner scanner = new Scanner(System.*in*);  
 while (true) {  
 System.*out*.println("Please choose an action:\n" +  
 "\n1 to see all numbers;" +  
 "\n2 to add a new number(10 numbers);" +  
 "\n3 to see a number of duplicates" +  
 "\n0 to exit;");  
 int num = scanner.nextInt();  
 switch (num) {  
 case 1 :{  
 System.*out*.print("| ");  
 for (Integer number : numbers) {  
 System.*out*.print(number + " ");  
 }  
 System.*out*.println("|");  
 break;  
 }  
 case 2 : {  
 String buffer = scanner.nextLine();  
 if (numbers.size() >= 10) {  
 System.*out*.println("There are already ten numbers, please choose other option:");  
 break;  
 }  
 System.*out*.println("Please enter a number to add:");  
 Integer number = scanner.nextInt();  
  
 numbers.add(number);  
 break;  
 }  
 case 3 : {  
 System.*out*.println(numbers.size() - new HashSet<>(numbers).size());  
 break;  
 }  
 case 0 : {  
 System.*exit*(0);  
 break;  
 }  
 default:  
 System.*out*.println("Please, enter a valid number.");  
 break;  
 }  
 }  
 }  
}

**Завдання №5:**

Создайте приложение для просмотра списка книг и удаления книг в библиотечном каталоге. Записи в списке (5 записей) являются объектами класса HashMap, где ключом является индекс ISBN книги (типа Integer), а значением – объект Book, содержащий наименование книги, фамилию, имя и отчество (ФИО) автора, издательство (все поля типа String), год издания (типа int) и цену книги (типа float). Предусмотреть возможность сортировки книг по 2-3 полям.

**Код програми:**

package task2;  
  
import java.util.\*;  
  
public class Task\_2 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Map<Integer, Book> library = new HashMap<>();  
 library.put(1, new Book("Six of Crows", "Leigh Bardugo", "Orion", 2015, 465));  
 library.put(2, new Book("The Restaurant at the End of the Universe", "Douglas Adams", "Pan Books", 1980, 300));  
 library.put(3, new Book("A Conjuring of Light", "V. E. Schwab", "Titan Books", 2017, 468));  
 library.put(4, new Book("The Lord of the Rings", "John Ronald Reuel Tolkien", "HarperCollins Publishers", 2014, 2703));  
 library.put(5, new Book("The Smoke Thieves", "Sally Green", "Penguin", 2018, 393));  
// System.out.println( new Book("Six of Crows", "Leigh Bardugo", "Orion", 2015, 465));  
 while (true) {  
 System.*out*.println("Please choose an action:\n" +  
 "\n1 to see the list of books;" +  
 "\n2 to delete a book by its ISBN;" +  
 "\n3 to see ordered list of books;" +  
 "\n0 to exit;");  
 int num = scanner.nextInt();  
 switch (num) {  
 case 1 : {  
 for (Map.Entry<Integer, Book> entry : library.entrySet()) {  
 System.*out*.println("ISBN: " + entry.getKey() + " | " + entry.getValue());  
 }  
 break;  
 }  
 case 2 : {  
 System.*out*.println("Please enter ISBN of a book you want to delete:");  
 int isbn = scanner.nextInt();  
 library.remove(isbn);  
 break;  
 }  
 case 3 : {  
 System.*out*.println("Please choose how would you like to sort books:\n" +  
 "\n1 By their name;" +  
 "\n2 By the name of author" +  
 "\n3 By their price" +  
 "\n0 to exit;");  
 int option = scanner.nextInt();  
 ArrayList<Book> books = new ArrayList<>(library.values());  
 switch (option) {  
 case 1:  
 books.sort(Comparator.*comparing*(Book::getName));  
 break;  
 case 2:  
 books.sort(Comparator.*comparing*(Book::getAuthorName));  
 break;  
 case 3:  
 books.sort(Comparator.*comparing*(Book::getPrice));  
 break;  
 case 0:  
 break;  
 default:  
 System.*out*.println("You've chosen an invalid option.");  
 }  
 for (Book book : books) {  
 System.*out*.println(book);  
 }  
 break;  
 }  
 case 0 : {  
 System.*exit*(0);  
 break;  
 }  
 default:  
 System.*out*.println("Please, enter a number which is present in menu.");  
 break;  
 }  
 }  
 }  
}

package task2;  
  
public class Book {  
  
 private String name;  
 private String authorName;  
 private String publisher;  
 private int publishingYear;  
 private float price;  
  
 public Book(String name, String authorName, String publisher, int publishingYear, float price) {  
 this.name = name;  
 this.authorName = authorName;  
 this.publisher = publisher;  
 this.publishingYear = publishingYear;  
 this.price = price;  
 }  
  
 public Book() {  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAuthorName() {  
 return authorName;  
 }  
  
 public void setAuthorName(String authorName) {  
 this.authorName = authorName;  
 }  
  
 public String getPublisher() {  
 return publisher;  
 }  
  
 public void setPublisher(String publisher) {  
 this.publisher = publisher;  
 }  
  
 public int getPublishingYear() {  
 return publishingYear;  
 }  
  
 public void setPublishingYear(int publishingYear) {  
 this.publishingYear = publishingYear;  
 }  
  
 public float getPrice() {  
 return price;  
 }  
  
 public void setPrice(float price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "Book: " +  
 "name='" + name + '\'' +  
 ", authorName='" + authorName + '\'' +  
 ", publisher='" + publisher + '\'' +  
 ", publishingYear=" + publishingYear +  
 ", price=" + price;  
 }  
}

**Unity Test:**

package task2;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class BookTest {  
 Book TestObject = new Book();  
 Book book1 = new Book("Six of Crows", "Leigh Bardugo", "Orion", 2015, 465);  
 private final ByteArrayOutputStream output = new ByteArrayOutputStream();  
 @BeforeEach  
 void setUp() {  
 System.*setOut*(new PrintStream(output));  
 }  
  
 @Test  
 void get\_setName() {  
 TestObject.setName("Six of Crows");  
 *assertEquals*(TestObject.getName(), "Six of Crows");  
 }  
  
 @Test  
 void get\_setAuthorName() {  
 TestObject.setAuthorName("Leigh Bardugo");  
 *assertEquals*(TestObject.getAuthorName(), "Leigh Bardugo");  
 }  
  
 @Test  
 void get\_setPublisher() {  
 TestObject.setPublisher("Orion");  
 *assertEquals*(TestObject.getPublisher(), "Orion");  
 }  
  
  
 @Test  
 void get\_setPublishingYear() {  
 TestObject.setPublishingYear(2015);  
 *assertEquals*(TestObject.getPublishingYear(), 2015);  
 }  
  
 @Test  
 void get\_setPrice() {  
 TestObject.setPrice(465);  
 *assertEquals*(TestObject.getPrice(), 465);  
 }  
  
 @Test  
 void testToString() {  
 String actual = "Book: name='Six of Crows', authorName='Leigh Bardugo', publisher='Orion', publishingYear=2015, price=465.0";  
 *assertEquals*(actual, book1.toString());  
 }  
}

**Завдання №**19**:**

Создайте приложение для добавления очередника в очередь и просмотра очереди. Список (5 записей) создается в программе и является объектом класса LinkedList. Запись списка является объектом QueuePerson, содержащим поля фамилии, имени и отчества очередника (все поля типа String), поле типа AddressValue для адреса очередника и очередника (типа int). В свою очередь, поле типа AddressValue содержит три поля типа String: города, наименование улицы и номер дома, а также поле типа int – номер квартиры (если номер квартиры равен 0, дом, в котором проживает субъект, имеет квартир). Записи в очереди сортируются в соответствии с приоритетом, и очередник добавляется последним в очереди своего приоритета. Предусмотреть возможность расчета количества очередников каждого приоритета, используя HashMap.

**Код програми:**

package task3;  
  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.LinkedList;  
import java.util.Map;  
  
public class Task\_3 {  
 public static LinkedList<QueuePerson> addPersonToQueue(QueuePerson person, LinkedList<QueuePerson> queue) {  
 int counter = 0;  
 Iterator<QueuePerson> iterator = queue.iterator();  
 while (iterator.hasNext() && person.getQueue() >= iterator.next().getQueue()) {  
 ++counter;  
 }  
 queue.add(counter, person);  
 return queue;  
 }  
  
 public static HashMap<Integer, Integer> countPriorities(LinkedList<QueuePerson> queuePersonList) {  
 HashMap<Integer, Integer> counter = new HashMap<>();  
 for (QueuePerson person : queuePersonList) {  
 int key = person.getQueue();  
 if (!counter.containsKey(key)) {  
 counter.put(key, 1);  
 }  
 else {  
 counter.put(key, counter.get(key) + 1);  
 }  
 }  
 return counter;  
 }  
 public static void PrintingQueue(LinkedList<QueuePerson> list){  
 for (QueuePerson person : list) {  
 System.*out*.println(person);  
 }  
 }  
 public static void PrintingPriorities(LinkedList<QueuePerson> list){  
 for (Map.Entry<Integer, Integer> entry: *countPriorities*(list).entrySet()) {  
 System.*out*.printf("%d priority : %d people%n", entry.getKey(), entry.getValue());  
 }  
 }  
 public static void main(String[] args) {  
 LinkedList<QueuePerson> queuePersonList = new LinkedList<>();  
  
 queuePersonList = *addPersonToQueue*(new QueuePerson("Ivanov", "Ivan", "Ivanovich",  
 new AddressValue("Kyiv","Yanhelia","5",0),1) , queuePersonList);  
 queuePersonList = *addPersonToQueue*(new QueuePerson("Petrov", "Petro", "Petrovich",  
 new AddressValue("Kyiv","Yanhelia","7",0),2) , queuePersonList);  
 queuePersonList = *addPersonToQueue*(new QueuePerson("Lukinskyi", "Dymitr", "Dmytrovych",  
 new AddressValue("Kyiv","Yanhelia","5",0),3) , queuePersonList);  
 queuePersonList = *addPersonToQueue*(new QueuePerson("Kovalskyi", "Andriy", "Oleksandrovych",  
 new AddressValue("Kyiv","Yanhelia","5",0),1) , queuePersonList);  
 queuePersonList = *addPersonToQueue*(new QueuePerson("Koval", "Pavlo", "Ivanovich",  
 new AddressValue("Kyiv","Yanhelia","5",0),2) , queuePersonList);  
  
 *PrintingQueue*(queuePersonList);  
 *PrintingPriorities*(queuePersonList);  
  
 }  
}

package task3;  
  
import java.util.Objects;  
  
public class QueuePerson {  
  
 private String surname;  
 private String name;  
 private String middleName;  
  
 private AddressValue address;  
 private int queue;  
  
 public QueuePerson(String surname, String name, String middleName, AddressValue address, int queue) {  
 this.surname = surname;  
 this.name = name;  
 this.middleName = middleName;  
 this.address = address;  
 this.queue = queue;  
 }  
  
 public QueuePerson() {  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getMiddleName() {  
 return middleName;  
 }  
  
 public void setMiddleName(String middleName) {  
 this.middleName = middleName;  
 }  
  
 public AddressValue getAddress() {  
 return address;  
 }  
  
 public void setAddress(AddressValue address) {  
 this.address = address;  
 }  
  
 public int getQueue() {  
 return queue;  
 }  
  
 public void setQueue(int queue) {  
 this.queue = queue;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof QueuePerson)) return false;  
 QueuePerson that = (QueuePerson) o;  
 return getQueue() == that.getQueue() && getSurname().equals(that.getSurname()) && getName().equals(that.getName()) && getMiddleName().equals(that.getMiddleName()) && getAddress().equals(that.getAddress());  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getSurname(), getName(), getMiddleName(), getAddress(), getQueue());  
 }  
  
  
 @Override  
 public String toString() {  
 return "QueuePerson{" +  
 "surname='" + surname + '\'' +  
 ", name='" + name + '\'' +  
 ", middleName='" + middleName + '\'' +  
 ", address=" + address +  
 ", queue=" + queue +  
 '}';  
 }  
}

package task3;  
  
import java.util.Objects;  
  
public class AddressValue {  
 private String city;  
 private String street;  
 private String streetNumber;  
 private int apartmentNumber;  
  
 public AddressValue(String city, String street, String streetNumber, int apartmentNumber) {  
 this.city = city;  
 this.street = street;  
 this.streetNumber = streetNumber;  
 this.apartmentNumber = apartmentNumber;  
 }  
  
 public AddressValue() {  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public void setStreet(String street) {  
 this.street = street;  
 }  
  
 public String getStreetNumber() {  
 return streetNumber;  
 }  
  
 public void setStreetNumber(String streetNumber) {  
 this.streetNumber = streetNumber;  
 }  
  
 public int getApartmentNumber() {  
 return apartmentNumber;  
 }  
  
 public void setApartmentNumber(int apartmentNumber) {  
 this.apartmentNumber = apartmentNumber;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof AddressValue)) return false;  
 AddressValue that = (AddressValue) o;  
 return getApartmentNumber() == that.getApartmentNumber() && getCity().equals(that.getCity()) && getStreet().equals(that.getStreet()) && getStreetNumber().equals(that.getStreetNumber());  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getCity(), getStreet(), getStreetNumber(), getApartmentNumber());  
 }  
  
 @Override  
 public String toString() {  
 return "AddressValue{" +  
 "city='" + city + '\'' +  
 ", street='" + street + '\'' +  
 ", streetNumber='" + streetNumber + '\'' +  
 ", apartmentNumber=" + apartmentNumber +  
 '}';  
 }  
}

**Unity Test:**

package task3;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
import java.util.HashMap;  
import java.util.LinkedList;  
  
import static junit.framework.TestCase.*assertEquals*;  
  
 class Task\_3Test {  
  
 Task\_3 TestObject = new Task\_3();  
 private final ByteArrayOutputStream output = new ByteArrayOutputStream();  
 LinkedList<QueuePerson> actual\_list = new LinkedList<>();  
 HashMap<Integer, Integer> actual\_map = new HashMap<>();  
  
 @BeforeEach  
 void setUp() {  
 actual\_list.add(new QueuePerson("Ivanov", "Ivan", "Ivanovich",  
 new AddressValue("Kyiv","Yanhelia","5",0),1));  
 actual\_list.add(new QueuePerson("Petrov", "Petro", "Petrovich",  
 new AddressValue("Kyiv","Yanhelia","7",0),2));  
 actual\_list.add(new QueuePerson("Lukinskyi", "Dymitr", "Dmytrovych",  
 new AddressValue("Kyiv","Yanhelia","5",0),3));  
 actual\_list.add(new QueuePerson("Kovalskyi", "Andriy", "Oleksandrovych",  
 new AddressValue("Kyiv","Yanhelia","5",0),1));  
 actual\_list.add(new QueuePerson("Koval", "Pavlo", "Ivanovich",  
 new AddressValue("Kyiv","Yanhelia","5",0),2));  
  
 actual\_map.put(1, 2);  
 actual\_map.put(2, 2);  
 actual\_map.put(3, 1);  
  
 System.*setOut*(new PrintStream(output));  
 }  
  
 @Test  
 void addPersonToQueue() {  
 LinkedList<QueuePerson> expected = new LinkedList<>();  
  
 expected = TestObject.*addPersonToQueue*(new QueuePerson("Ivanov", "Ivan", "Ivanovich",  
 new AddressValue("Kyiv","Yanhelia","5",0),1) , expected);  
 expected = TestObject.*addPersonToQueue*(new QueuePerson("Petrov", "Petro", "Petrovich",  
 new AddressValue("Kyiv","Yanhelia","7",0),2) , expected);  
 expected = TestObject.*addPersonToQueue*(new QueuePerson("Lukinskyi", "Dymitr", "Dmytrovych",  
 new AddressValue("Kyiv","Yanhelia","5",0),3) , expected);  
 expected = TestObject.*addPersonToQueue*(new QueuePerson("Kovalskyi", "Andriy", "Oleksandrovych",  
 new AddressValue("Kyiv","Yanhelia","5",0),1) , expected);  
 expected = TestObject.*addPersonToQueue*(new QueuePerson("Koval", "Pavlo", "Ivanovich",  
 new AddressValue("Kyiv","Yanhelia","5",0),2) , expected);  
  
 *assertEquals*(expected.get(0), actual\_list.get(0));  
 *assertEquals*(expected.getClass().getSimpleName(), actual\_list.getClass().getSimpleName());  
 }  
}

package task3;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class QueuePersonTest {  
 QueuePerson TestObject = new QueuePerson();  
  
 @Test  
 void getSurname() {  
 TestObject.setSurname("Ivanov");  
 *assertEquals*(TestObject.getSurname(), "Ivanov");  
 }  
  
 @Test  
 void setSurname() {  
 TestObject.setName("Ivan");  
 *assertEquals*(TestObject.getName(), "Ivan");  
 }  
  
 @Test  
 void getName() {  
 TestObject.setName("Ivan");  
 *assertEquals*(TestObject.getName(), "Ivan");  
 }  
  
 @Test  
 void setName() {  
 TestObject.setSurname("Ivanov");  
 *assertEquals*(TestObject.getSurname(), "Ivanov");  
 }  
  
 @Test  
 void getMiddleName() {  
 TestObject.setMiddleName("Ivanovich");  
 *assertEquals*(TestObject.getMiddleName(), "Ivanovich");  
 }  
  
 @Test  
 void setMiddleName() {  
 TestObject.setMiddleName("Ivanovich");  
 *assertEquals*(TestObject.getMiddleName(), "Ivanovich");  
 }  
  
// @Test  
// void getAddress() {  
// TestObject.setAddress(new AddressValue("Kyiv","Yanhelia","5",0));  
// assertEquals(TestObject.getAddress(),  
// "AddressValue{city='Kyiv', street='Yanhelia', streetNumber='5', apartmentNumber=0}");  
// }  
//  
// @Test  
// void setAddress() {  
// TestObject.setAddress(new AddressValue("Kyiv","Yanhelia","5",0));  
// assertEquals(TestObject.getAddress(),  
// "AddressValue{city='Kyiv', street='Yanhelia', streetNumber='5', apartmentNumber=0}");  
// }  
  
 @Test  
 void getQueue() {  
 TestObject.setQueue(1);  
 *assertEquals*(TestObject.getQueue(), 1);  
 }  
  
 @Test  
 void setQueue() {  
 TestObject.setQueue(1);  
 *assertEquals*(TestObject.getQueue(), 1);  
 }  
}

package task3;  
  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
class AddressValueTest {  
  
 AddressValue TestObject = new AddressValue();  
  
 @Test  
 void getCity() {  
 TestObject.setCity("Kyiv");  
 *assertEquals*(TestObject.getCity(), "Kyiv");  
 }  
  
 @Test  
 void setCity() {  
 TestObject.setCity("Kyiv");  
 *assertEquals*(TestObject.getCity(), "Kyiv");  
  
 }  
 @Test  
 void getStreet() {  
 TestObject.setStreet("Yanhelia");  
 *assertEquals*(TestObject.getStreet(), "Yanhelia");  
 }  
  
 @Test  
 void setStreet() {  
 TestObject.setStreet("Yanhelia");  
 *assertEquals*(TestObject.getStreet(), "Yanhelia");  
 }  
  
 @Test  
 void getStreetNumber() {  
 TestObject.setStreetNumber("5");  
 *assertEquals*(TestObject.getStreetNumber(), "5");  
 }  
  
 @Test  
 void setStreetNumber() {  
 TestObject.setStreetNumber("5");  
 *assertEquals*(TestObject.getStreetNumber(), "5");  
 }  
  
 @Test  
 void getApartmentNumber() {  
 TestObject.setApartmentNumber(0);  
 *assertEquals*(TestObject.getApartmentNumber(), 0);  
 }  
  
 @Test  
 void setApartmentNumber() {  
 TestObject.setApartmentNumber(0);  
 *assertEquals*(TestObject.getApartmentNumber(), 0);  
 }  
}

**Завдання №29:**

Создайте приложение для просмотра списка файлов добавления файла в список. Список (для 5 текстовых файлов с расширением .txt) создается программе и является объектом класса HashMap, ключом является имя файла (типа String), а значением – объект типа TextFile, содержащий элемента String – абсолютный путь к файлу (без имени файла) и краткое описание содержимого файла.

**Код програми:**

package task4;  
  
import java.util.\*;  
  
public class Task\_4 {  
 public static void main(String[] args) {  
  
 HashMap<String,TextFile> files = new HashMap<>();  
 files.put("song.txt", new TextFile("C:\\Program Files", "Song lyrics"));  
 files.put("lab.txt", new TextFile("C:\\my\_tasks\\homework\\","Task description"));  
  
 System.*out*.println(new TextFile("C:\\Program Files", "Song lyrics"));  
  
 Scanner scanner = new Scanner(System.*in*);  
 while (true) {  
 System.*out*.println("Please choose an action:\n" +  
 "\n1 to see all files;" +  
 "\n2 to add a new file;" +  
 "\n0 to exit;");  
 int num = scanner.nextInt();  
 switch (num) {  
 case 1 : {  
 for (Map.Entry<String, TextFile> entry : files.entrySet()) {  
 System.*out*.println("File: " + entry.getKey() + " " + entry.getValue());  
 }  
 break;  
 }  
 case 2 : {  
 String buffer = scanner.nextLine();  
 if (files.size() >= 5) {  
 System.*out*.println("There are already five files stored, please choose other option:");  
 break;  
 }  
 System.*out*.println("Please enter a file name:");  
 String fileName = scanner.nextLine();  
  
 System.*out*.println("Please enter a file's absolute path:");  
 String filePath = scanner.nextLine();  
  
 System.*out*.println("Please enter a file's description:");  
 String fileDescription = scanner.nextLine();  
 files.put(fileName, new TextFile(filePath, fileDescription));  
 break;  
 }  
 case 0 : {  
 System.*exit*(0);  
 break;  
 }  
 default:  
 System.*out*.println("Please, enter a valid number.");  
 break;  
 }  
 }  
 }  
}

package task4;  
  
public class TextFile {  
  
 private String absolutePath;  
 private String description;  
  
 public TextFile(String absolutePath, String description) {  
 this.absolutePath = absolutePath;  
 this.description = description;  
 }  
  
 public TextFile() {  
 }  
  
 public String getAbsolutePath() {  
 return absolutePath;  
 }  
  
 public void setAbsolutePath(String absolutePath) {  
 this.absolutePath = absolutePath;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 @Override  
 public String toString() {  
 return "TextFile{" +  
 "absolutePath='" + absolutePath + '\'' +  
 ", description='" + description + '\'' +  
 '}';  
 }  
}

**Unity Test:**

package task4;  
  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TextFileTest {  
 TextFile TestObject = new TextFile();  
 TextFile file1 = new TextFile("C:\\Program Files", "Song lyrics");  
 @Test  
 void get\_setAbsolutePath() {  
 TestObject.setAbsolutePath("C:\\Program Files");  
 *assertEquals*(TestObject.getAbsolutePath(), "C:\\Program Files");  
 }  
  
 @Test  
 void get\_setDescription() {  
 TestObject.setDescription("Song lyrics");  
 *assertEquals*(TestObject.getDescription(), "Song lyrics");  
 }  
  
 @Test  
 void testToString() {  
 String actual = "TextFile{absolutePath='C:\\Program Files', description='Song lyrics'}";  
 *assertEquals*(actual, file1.toString());  
 }  
}