МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«Харківський Політехнічний Інститут»  
  
Кафедра «Стратегічного Управління»

ЗВІТ

з лабораторної роботи № 2

«Робота з винятками і файлами в Java»

Перевірила: старший викладач  
Вільхівська О. В.  
Виконав: ст. гр. КН-27

Харків, 2018

Лабораторна робота № 2  
Тема: Робота з винятками і файлами в Java  
Завдання

### Індивідуальне завдання

Спроектувати та реалізувати класи для представлення сутностей попередньої лабораторної роботи. Рішення повинне базуватися на раніше створеній ієрархії класів. Слід створити два похідних класи від класу, який представляє основну сутність. Один клас повинен бути доповненим можливостями читання даних з відповідно підготовленого текстового файлу та запису цих даних в інший файл після сортування. Другий клас повинен реалізовувати читання даних з XML-документу, зберігати дані в структурах, які автоматично створюються за допомогою технології зв'язування даних, та запис даних в інший XML-документ після сортування. Для уникнення дублювання даних у програмі слід також перевизначити клас, який представляє другу сутність. Похідні класи, які представляють основну сутність, повинні реалізовувати спільний інтерфейс, в якому оголошені функції читання з файлу і запису в файл.  
  
Окрім роботи з файлами повинно бути реалізоване виведення результатів у консольне вікно.

### Сортування цілих

Реалізувати програму читання з текстового файлу цілих додатних значень (числа розділені пробілами, слід читати до кінця файлу), занесення цих чисел у масив, сортування за зменшенням та за збільшенням суми цифр та зберігання обох результатів у двох нових текстових файлах. Перелічені дії реалізувати в окремій статичній функції. Для визначення порядку сортування створити класи, які реалізують інтерфейс Comparator.

### Реалізація серіалізації й десеріалізації

Описати класи Студент і Академічна група (з полем - масивом студентів). Створити об'єкти, здійснити їх бінарну серіалізацію й десеріалізацію, а також серіалізацію й десеріалізацію в XML.

### Список файлів усіх підкаталогів

Увести з клавіатури ім'я певної теки. Вивести на екран імена усіх файлів цієї теки, а також усіх файлів підкаталогів, їхніх підкаталогів тощо. Реалізувати виведення через рекурсивну функцію. Якщо тека не існує, вивести повідомлення про помилку.

### Робота з ZIP-архівом (додаткове завдання)

Описати класи Студент і Академічна група (з полем - масивом студентів). Створити об'єкти, здійснити запис даних про студентів академічної групи в архів. В іншій програмі здійснити читання з архіву.

Хід роботи

### Завдання 1. Індивідуальне завдання

*Код програми*

*Вміст AbstractSubject.java*

|  |
| --- |
| package task\_1;  import java.util.\*;  public abstract class AbstractSubject {  public abstract void setTopics(AbstractTopic[] topics);  public abstract void setName(String str);  public abstract void sortByTopicLength();  public abstract void sortByStudents();  public abstract void ClearTopics();    public abstract String getName();  public abstract AbstractTopic[] getTopics();  public abstract boolean getExams();  public abstract void setExams(boolean b);  // public abstract boolean addTopic(AbstractTopic topic);  public double getAverage() {  double average=0;  for (AbstractTopic topic: getTopics()) {  average+=topic.getAmount();  }  return average/=getTopics().length;  }    public static AbstractTopic[] addToArray(AbstractTopic[] arr, AbstractTopic item) {  AbstractTopic[] newArr;  if (arr!=null) {  newArr = new AbstractTopic[arr.length+1];  System.arraycopy(arr, 0, newArr, 0, arr.length);  }  else {  newArr = new AbstractTopic[1];  }  newArr[newArr.length-1]=item;  return newArr;  }  public AbstractTopic getMin() {  AbstractTopic lecture = getTopics()[0];  for (int i=1; i<getTopics().length; i++) {  if(getTopics()[i].getAmount()>lecture.getAmount()) {  lecture = getTopics()[i];  }  }  return lecture;  }  public AbstractTopic[] findWord(String word) {  System.out.println("\nInfo about lecture containing word " + word);  AbstractTopic[] at = {};  for (AbstractTopic topic: getTopics()) {  if (topic.containsWord(word)) {  System.out.println(topic);  addToArray(at,topic);  }  }  return at; // Возвращает массив с найденными лекциями  }  @Override  public String toString() { //overloading toString() method  String result = "Name: " + getName() + " Exams: " + getExams() + " Amount of practices: " + getTopics().length;  for (int i = 0; i < getTopics().length; i++) {  result += "\n" + getTopics()[i]; // Раньше был метод, возращающий конкретный элемент  }  return result;  }  @Override  public boolean equals(Object obj) {  if (this==obj) return true;  if ((obj==null)||!(obj instanceof AbstractSubject)) return false;  AbstractSubject as = (AbstractSubject) obj;  if (!getName().equals(as.getName())) return false;  return Arrays.equals(getTopics(), as.getTopics());    }  public boolean addTopic(AbstractTopic topic) {  if (getTopics()!=null) {  for (AbstractTopic p : getTopics()) {  if (p.equals(topic)) {  return false;  }  }  }  setTopics(addToArray(getTopics(), topic));  return true;  }  protected AbstractSubject createSubject() {  setName("OOP");  System.out.println("\nAdding new practices to " + getName());  System.out.println(addTopic(new TopicWithData("21,9,2018", "Generics", 8)));  System.out.println(addTopic(new TopicWithData("2,10,2018", "Exceptions", 3)));  System.out.println(addTopic(new TopicWithData("16,10,2018", "GUI", 8)));  System.out.println(addTopic(new TopicWithData("30,10,2018", "Reflection", 7)));  System.out.println(addTopic(new TopicWithData("30,10,2018", "Reflection", 7)));  return this;  }  public void testSubject() {  sortByStudents(); // Сортирует по количеству студентов на паре  System.out.println("\nSorting by students amount: \n" + this); // С помощью перегрузки toString() выводит всю инфу о списке лекций  sortByTopicLength(); // Сортирует по длине названия лекции  System.out.println("\nSorting by the length of a topic name: \n" + this);  findWord("Reflection"); // Пишет инфу о предметах с найденным словом  }  public int topicsCount() {  // TODO Auto-generated method stub  return 0;  }  } |

*Вміст AbstractTopic.java*

|  |
| --- |
| package task\_1;  import java.util.\*;  public abstract class AbstractTopic implements Comparable<AbstractTopic>{  public abstract void setName(String name);  public abstract void setAmount(int amount);    public abstract String getDate();  public abstract String getName();  public abstract int getAmount();  public boolean containsWord(String word) {  StringTokenizer st = new StringTokenizer(getName());  String s;  while (st.hasMoreTokens()) {  s = st.nextToken();  if (s.toLowerCase().equals(word.toLowerCase())) {  return true;  }  }  return false;  }  @Override  public int compareTo (AbstractTopic at) {  return Integer.compare(getAmount(), at.getAmount());  }  @Override  public boolean equals(Object obj) {  if (this == obj)  return true;  if (obj==null || !(obj instanceof TopicWithData))  return false;  TopicWithData test = (TopicWithData) obj;  return (test.getDate().equals(getDate())) &&  test.getAmount() == getAmount() &&  test.getName() == getName();    }  @Override  public String toString() { //overloading toString() method  return ("Date: " + getDate() + " Subject: " + getName() + " Students: " + getAmount());  }  protected void testTopic() {  setDate("14,12,2018");  setName("Collections");  setAmount(5);  }  public void setDate(String date) {}  } |

*Вміст FileIO.java*

|  |
| --- |
| package task\_1.WriteToFile;  public interface FileIO {  void readFromFile(String fileName) throws Exception;  void writeToFile(String fileName) throws Exception;  } |

*Вміст TextFileSubject.java*

|  |
| --- |
| package task\_1.WriteToFile;  import java.io.\*;  import java.util.Arrays;  import java.util.InputMismatchException;  import java.util.Scanner;  import task\_1.AbstractTopic;  import task\_1.SubjectWithArray;  import task\_1.TopicWithData;  public class TextFileSubject extends SubjectWithArray implements FileIO {  @Override  public void readFromFile(String fileName) throws FileNotFoundException, InputMismatchException {  try (Scanner scanner = new Scanner(new FileReader(fileName))) {  setName(scanner.next());  setExams(scanner.nextBoolean());    while (scanner.hasNext()) {  addTopic(new TopicWithData(scanner.next(), scanner.next(),scanner.nextInt()));  }  }  }  @Override  public void writeToFile(String fileName) throws IOException {  try (PrintWriter out = new PrintWriter(new FileWriter(fileName))) {  out.println(getName() + " " + getExams());  for (AbstractTopic topic : getTopics()) {  out.println(topic.getDate() + " " + topic.getName() + " " + topic.getAmount());  }  }  }    public static void main(String[] args) {  TextFileSubject subject = new TextFileSubject();  // subject.createSubject();  try {  subject.readFromFile("src/task\_1/OOP.txt");  subject.testSubject();  subject.writeToFile("src/task\_1/ByNames.txt");  }  catch (FileNotFoundException e) {  System.err.println("Read failed");  e.printStackTrace();  }  catch (IOException e) {  System.err.println("Write failed");  e.printStackTrace();  }  catch (InputMismatchException e) {  e.printStackTrace();  System.err.println("Wrong format");  }  }  } |

*Приклад роботи методів класу TextFileSubject:*

*Вміст вихідного файлу OOP.txt (який створюємо власноруч)*

|  |
| --- |
| *OOP false*  *21,9,2018 Generics 8*  *30,10,2018 Reflection 7*  *2,10,2018 Exceptions 3*  *16,10,2018 GUI 8* |

*Вміст похідного файлу ByNames.txt (що був записаний після виклику функції writeToFile класу TextFileSubject). Записи відсортовано за довжиною назви тем*

|  |
| --- |
| *OOP false*  *16,10,2018 GUI 8*  *21,9,2018 Generics 8*  *2,10,2018 Exceptions 3*  *30,10,2018 Reflection 7* |

*Вміст Subject.xsd*

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  <xs:element name="SubjectData">  <xs:complexType>  <xs:sequence>  <xs:element maxOccurs="unbounded" name="TopicData">  <xs:complexType>  <xs:attribute name="Name" type="xs:string" use="required" />  <xs:attribute name="Date" type="xs:string" use="required" />  <xs:attribute name="Amount" type="xs:int" use="required" />  </xs:complexType>  </xs:element>  </xs:sequence>  <xs:attribute name="Name" type="xs:string" use="required" />  <xs:attribute name="Exams" type="xs:boolean" use="required" />  </xs:complexType>  </xs:element>  </xs:schema> |

*Вміст XMLTopic.java*

|  |
| --- |
| package task\_1.xml;  import task\_1.xml.SubjectData;  import task\_1.xml.SubjectData.TopicData;  import task\_1.AbstractTopic;  public class XMLTopic extends AbstractTopic{  SubjectData.TopicData topicData;    public XMLTopic (SubjectData.TopicData topicData){  this.topicData = topicData;  }  @Override  public void setDate(String date) {  topicData.setDate(date);  }  @Override  public void setName(String name) {  topicData.setName(name);  }  @Override  public void setAmount(int amount) {  topicData.setAmount(amount);;  }  @Override  public String getDate() {  return topicData.getDate();  }  @Override  public String getName() {  return topicData.getName();  }  @Override  public int getAmount() {  return topicData.getAmount();  }  } |

*Вміст XMLSubject.java*

|  |
| --- |
| package task\_1.xml;  import task\_1.AbstractTopic;  import task\_1.AbstractSubject;  import task\_1.xml.SubjectData;  import task\_1.xml.SubjectData.TopicData;  import task\_1.WriteToFile.FileIO;  import javax.xml.bind.JAXBContext;  import javax.xml.bind.JAXBException;  import javax.xml.bind.Marshaller;  import javax.xml.bind.Unmarshaller;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.FileWriter;  import java.io.IOException;  import java.util.Collections;  import java.util.Comparator;  public class XMLSubject extends AbstractSubject implements FileIO {  private SubjectData subjectData = new SubjectData();    @Override  public void readFromFile(String fileName) throws JAXBException, FileNotFoundException {  JAXBContext jaxbContext = JAXBContext.newInstance("task\_1");  Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();  subjectData = (SubjectData) unmarshaller.unmarshal(new FileInputStream(fileName));  }  @Override  public void writeToFile(String fileName) throws Exception {  JAXBContext jaxbContext = JAXBContext.newInstance("task\_1");  Marshaller marshaller = jaxbContext.createMarshaller();  marshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, Boolean.TRUE);  marshaller.marshal(subjectData, new FileWriter(fileName));  }  @Override  public void ClearTopics() {  subjectData.getTopicData().clear();  }  @Override  public void setName(String str) {  subjectData.setName(str);  }  public SubjectData.TopicData getTopicData(int i){  return subjectData.getTopicData().get(i);  }  @Override  public void sortByTopicLength() {  Collections.sort(subjectData.getTopicData(),  Comparator.comparing((t1)->t1.getName().length()));  }  @Override  public void sortByStudents() {  Collections.sort(subjectData.getTopicData(),  Comparator.comparing(SubjectData.TopicData::getAmount));  }  @Override  public String getName() {  return subjectData.getName();  }  @Override public int topicsCount() {  return subjectData.getTopicData().size();  }  @Override  public AbstractTopic[] getTopics() {  AbstractTopic[] topics = new AbstractTopic[topicsCount()];  for (int i = 0; i < topicsCount(); i++) {  topics[i] = new XMLTopic(subjectData.getTopicData().get(i));  }  return topics;  }  @Override  public boolean getExams() {  return subjectData.isExams();  }  @Override  public void setExams(boolean b) {  subjectData.setExams(b);  }  @Override  public void setTopics(AbstractTopic[] topics) {  ClearTopics();  for (AbstractTopic topic : topics) {  addTopic(topic);  }  }  public static void main(String[] args) throws Exception {  XMLSubject subject = new XMLSubject();  try {  subject.readFromFile("src/task\_1/xml/OOP.xml");  subject.testSubject();  subject.writeToFile("src/task\_1/xml/ByNames.xml");  }  catch (FileNotFoundException e) {  System.out.println("Read failed");  e.printStackTrace();  }  catch (IOException e) {  System.out.println("Write failed");  e.printStackTrace();  }  catch (JAXBException e) {  e.printStackTrace();  System.out.println("Wrong format");  }  }  } |

*Приклад роботи методів класу XMLSubject*

*Вміст вихідного OOP.xml необхідного для тестування зчитування даних*

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <SubjectData Name="OOP" Exams="false" >  <TopicData Date="21/9/2018" Name="Generics" Amount="8" />  <TopicData Date="2/10/2018" Name="Exceptions" Amount="3" />  <TopicData Date="16/10/2018" Name="GUI" Amount="8" />  <TopicData Date="30/10/2018" Name="Reflection" Amount="7" />  </SubjectData> |

*Вміст похідного ByNames.xml з відсотрованими значеннями*

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="yes"?>  <SubjectData Name="OOP" Exams="false">  <TopicData Name="GUI" Date="16/10/2018" Amount="8"/>  <TopicData Name="Generics" Date="21/9/2018" Amount="8"/>  <TopicData Name="Exceptions" Date="2/10/2018" Amount="3"/>  <TopicData Name="Reflection" Date="30/10/2018" Amount="7"/>  </SubjectData> |

### Завдання 2. Сортування цілих

*Код програми*

|  |
| --- |
| package task\_2; import java.io.\*; import java.util.\*;  public class task\_2{  public static Comparator<Integer> CompareIntegers(){  class LocalComparator implements Comparator<Integer>{  @Override  public int compare(Integer int1, Integer int2) {  int sum1=0, sum2=0;  while(int1>0){  sum1+=int1%10;  int1/=10;  }  while(int2>0){  sum2+=int2%10;  int2/=10;  }  return Integer.compare(sum1, sum2);  }  }  return new LocalComparator();  }   public static void sort(String inFileName, String outFileName\_1, String outFileName\_2) throws IOException, InputMismatchException{  Integer[] arr = {};  try (BufferedReader br = new BufferedReader(new FileReader (inFileName))){  Scanner scanner = new Scanner (br);  while (scanner.hasNext()){  Integer[] arr1 = new Integer[arr.length+1];  System.arraycopy(arr,0,arr1,0,arr.length);  arr1[arr.length]=scanner.nextInt();  arr=arr1;  }  }  PrintWriter pw\_1 = new PrintWriter(new FileWriter(outFileName\_1));  PrintWriter pw\_2 = new PrintWriter(new FileWriter(outFileName\_2));  try{  Arrays.sort(arr,CompareIntegers());  for (int i=0; i<arr.length; i++){  pw\_1.print(arr[i]+" ");  pw\_2.print(arr[arr.length-i-1]+" ");  }  }  finally{  pw\_1.close();  pw\_2.close();  }  }   public static void main (String[] args){  try{  sort("src/task\_2/data.txt", "src/task\_2/sorted\_1.txt","src/task\_2/sorted\_2.txt");  }  catch(IOException e){  e.printStackTrace();  }  catch(InputMismatchException e){  e.printStackTrace();  }  } } |

*Приклад виконання програми*

*Вміст data.txt*

|  |
| --- |
| 5 6 4 9 8 0 10 111 43 |

*Вміст sorted\_1.txt*

|  |
| --- |
| 0 10 111 4 5 6 43 8 9 |

*Вміст sorted\_2.txt*

|  |
| --- |
| 9 8 43 6 5 4 111 10 0 |

### Завдання 3. Реалізація серіалізації й десеріалізації

*Вміст Student.java*

|  |
| --- |
| package task\_3;  import java.io.Serializable;  public class Student implements Serializable {  private final static long serialVersiouUID = 2;  private String name;  private String sname;   public Student(){}; // To correspond Java Bean class requirements   public Student(String name, String sname) {  this.name = name;  this.sname = sname;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getSname() {  return sname;  }   public void setSname(String sname) {  this.sname = sname;  } } |

*Вміст group.java*

|  |
| --- |
| package task\_3;  import java.io.FileOutputStream; import java.io.IOException; import java.io.ObjectOutputStream; import java.io.Serializable;  public class group implements Serializable {  private final static long serialVersiouUID = 1;  private String name;  private Student[] students;   public group(){}; // To correspond Java Bean class requirements   public group (String name, Student[] students){  this.name = name;  this.students = students;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public Student[] getStudents() {  return students;  }   public void setStudents(Student[] students) {  this.students = students;  }  } |

*Вміст Serialization.java*

|  |
| --- |
| package task\_3;  import java.beans.XMLDecoder; import java.beans.XMLEncoder; import java.io.\*;  public class Serialization {  public static void main(String[] args) throws ClassNotFoundException{   group gr1 = new group("CS-27",new Student[]{new Student("Alex", "Bloodless"), new Student("Bogdan", "Dolgozvagov")});   //Serialization  try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("src/task\_3/out.dat"))){  out.writeObject(gr1);  }  catch(IOException e){  e.printStackTrace();  }   //Deserialization  try(ObjectInputStream in = new ObjectInputStream(new FileInputStream("src/task\_3/out.dat"))){  group gr2 = (group) in.readObject();  for (Student st : gr2.getStudents()) {  System.out.println(st.getName() + " " + st.getSname());  }  }  catch(IOException e){  e.printStackTrace();  }   //XML Serialization  try (XMLEncoder xmlEncoder = new XMLEncoder(new FileOutputStream("src/task\_3/object.xml"))){  xmlEncoder.writeObject(gr1);  xmlEncoder.flush();  }  catch(IOException e){  e.printStackTrace();  }   //XML Deserialization  try (XMLDecoder xmlDecoder = new XMLDecoder(new FileInputStream("src/task\_3/object.xml"))){  group gr3 = (group) xmlDecoder.readObject();  for (Student st : gr3.getStudents()) {  System.out.println(st.getName() + " " + st.getSname());  }  }  catch(IOException e){  e.printStackTrace();  }  } } |

*Приклад виконання програми*

|  |
| --- |
| Alex Bloodless Bogdan Dolgozvagov Alex Bloodless Bogdan Dolgozvagov |

*Вміст object.xml*

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <java version="1.8.0\_181" class="java.beans.XMLDecoder">  <object class="task\_3.group">  <void property="name">  <string>CS-27</string>  </void>  <void property="students">  <array class="task\_3.Student" length="2">  <void index="0">  <object class="task\_3.Student">  <void property="name">  <string>Alex</string>  </void>  <void property="sname">  <string>Bloodless</string>  </void>  </object>  </void>  <void index="1">  <object class="task\_3.Student">  <void property="name">  <string>Bogdan</string>  </void>  <void property="sname">  <string>Dolgozvagov</string>  </void>  </object>  </void>  </array>  </void>  </object> </java> |

### Завдання 4. Список файлів усіх підкаталогів

*Код програми*

|  |
| --- |
| package task\_4;  import java.io.\*; import java.nio.file.NotDirectoryException; import java.util.\*;  public class task\_4 {  public static String space = "\t";  public static int count;  public static void showFiles(File f) throws IOException, NullPointerException {  if (f.isDirectory()) {  for (File file : f.listFiles()) {  for (int i = 0; i < count; i++) {  System.out.print(space);  }  System.out.println(file);  count++;  showFiles(file);  count--;  }  }  }   public static void main(String[] args) throws IOException { // Scanner scanner = new Scanner(System.in); // String dirName = scanner.next();  String dirName = "src/task\_4/test";  // String dirName = "D:/Movies";  File dir = new File(dirName);  try {  showFiles(dir);  }  catch (IOException e){  e.printStackTrace();  }  catch (NullPointerException e){  //e.printStackTrace();  System.out.println("No such directory!");  }  } } |

*Приклад виконання програми*

|  |
| --- |
| src\task\_4\test\1  src\task\_4\test\1\1  src\task\_4\test\1\2  src\task\_4\test\1\2\1  src\task\_4\test\1\2\2  src\task\_4\test\1\2\3 src\task\_4\test\2  src\task\_4\test\2\1  src\task\_4\test\2\2 src\task\_4\test\3  src\task\_4\test\3\1 |

### Завдання 5. Робота з ZIP-архівом (додаткове завдання)

*Код програми*

*Вміст файлів Student.java та group.java збігається з однойменними файлами третього завдання.*

*Вміст Zipping.java*

|  |
| --- |
| package task\_5;  import java.io.\*; import java.util.zip.\*;  public class Zipping {  public static void main(String[] args) {   group gr1 = new group("CS-27",new Student[]{  new Student("Alex", "Bloodless"),  new Student("Bogdan", "Dolgozvagov")});   //Zipping  try(ZipOutputStream zOut = new ZipOutputStream(new FileOutputStream("src/task\_5/Group.zip"))){  DataOutputStream out = new DataOutputStream(zOut);  for (Student st : gr1.getStudents()){  ZipEntry zipEntry = new ZipEntry(st.getName());  zOut.putNextEntry(zipEntry);  out.writeChars(st.getName());  out.writeChars(st.getSname());  zOut.closeEntry();  }  }  catch (IOException e){  e.printStackTrace();  }   //Unzipping  try(ZipInputStream zIn = new ZipInputStream(new FileInputStream("src/task\_5/Group.zip"))){  //DataInputStream in = new DataInputStream(zIn);  InputStreamReader in = new InputStreamReader(zIn);  BufferedReader br = new BufferedReader(in);   ZipEntry zipEntry;  while ((zipEntry= zIn.getNextEntry())!=null){  System.out.println(zipEntry.getName()+" ");  System.out.println(br.readLine());  }  }  catch(IOException e){  e.printStackTrace();  }  } } |

*Приклад виконання програми*

|  |
| --- |
| Alex   A l e x B l o o d l e s s Bogdan   B o g d a n D o l g o z v a g o v |

Висновки

Класи, які здійснюють файлове введення та виведення, а також інші дії з потоками, розташовані у пакеті java.io. Класи цього пакету пропонують низку методів для створення таких потоків, читання, запису, тощо. Безпосередню роботу з текстовими файлами здійснюють об'єкти класів FileReader та FileWriter. Існує дві підмножини класів - відповідно для роботи з текстовими та бінарними (двійковими) файлами.

Розширювана мова розмічування XML (eXtensible Markup Language) - це незалежний від платформи метод структурування інформації. Оскільки XML відокремлює зміст документу від його структури, його успішно використовують для обміну інформацією. Наприклад, XML можна використовувати для передачі даних між програмою та базами даних, або між базами даних, що мають різні формати.