Лабораторна робота №  
Тема: Робота з узагальненнями та колекціями в Java  
Завдання

1. **Індивідуальне завдання**

Розробити ієрархію класів для представлення сутностей індивідуального завдання лабораторної роботи № 5 курсу "Алгоритмізація та програмування" (друга частина). Базовий абстрактний клас, який представляє другу сутність індивідуального завдання, не повинен містити даних, лише абстрактні методи доступу, перевизначення функцій toString() та equals(), а також реалізацію функцій, визначених попереднім завданням. Цей клас також повинен реалізовувати інтерфейс Comparable для природного порівняння об'єктів під час сортування за однією з ознак.  
  
Базовий абстрактний клас, який представляє першу з сутностей індивідуального завдання, повинен містити:

* абстрактні функції для доступу до даних;
* абстрактні функції для доступу до послідовності елементів типу другого абстрактного класу;
* абстрактні функції сортування елементів послідовності за визначеними ознаками відповідно до завдання лабораторної роботи № 6 курсу "Алгоритмізація та програмування" (друга частина);
* перевизначення функції toString() для виведення даних про об'єкти;
* перевизначення методу equals() для перевірки еквівалентності об'єктів;
* реалізацію методів пошуку за визначеними ознаками;
* реалізацію методіу тестувння функціональності класів.

Похідні класи від створених абстрактних класів повинні містити поля конкретних типів, зокрема, послідовність елементів другої сутності повинна бути представлена у різних похідних класах у вигляді масиву та списку.  
  
Здійснити тестування обох реалізацій. Тест повинен включати виконання завдання попередніх лабораторних робіт. Для сортування слід використовувати методи sort() класів Arrays та Collections відповідно. Для визначення другої ознаки сортування використати лямбда-вираз.

1. **Мінімум функції**

Реалізувати програму, що дозволяє знайти мінімум деякої функції на заданому інтервалі. Алгоритм знаходження мінімуму полягає в послідовному переборі з певним кроком точок інтервалу і порівнянні значень функції в поточній точці з раніше знайденим мінімумом.  
  
Знайти та застосувати відповідний стандартний функціональний інтерфейс для опису функції.  
  
Реалізувати три підходи - через використання безіменного класу, лямбда-виразів і вказівників на методи.

1. **Узагальнений клас**

Створити узагальнений клас для зберігання довільних даних у масиві. Реалізувати функцію додавання елементу в кінець масиву, видалення елементу, додавання групи (іншого масиву) елементів.

1. **Створення бібліотеки узагальнених функцій для роботи з масивами та списками**

Реалізувати клас зі статичними узагальненими методами, які реалізують таку функціональність:

* обмін місцями двох груп елементів
* обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом)
* вставлення у масив (список) іншого масиву (списку) елементів у вказане місце
* заміна групи елементів іншим масивом (списком) елементів

Реалізувати наведені функції для масивів і для списків. Здійснити демонстрацію роботи усіх методів з використанням даних різних типів (Integer, Double, String) .

1. **Реалізація інтерфейсу Comparable**

Створити клас Circle, який реалізує інтерфейс Comparable. Більшим вважається коло з більшим радіусом. Здійснити сортування списку об'єктів типу Circle.

1. **Реалізація інтерфейсу Comparator**

Створити клас Triangle. Трикутник визначати довжинами сторін. Площа трикутника в цьому випадку може бути обчислена за формулою Герона:



де a, b і c - довжини сторін трикутника. Здійснити сортування списку трикутників за зменшенням площі. Для визначення ознаки сортування використовувати об'єкт, який реалізує інтерфейс Comparator.

1. **Реалізація масиву точок за допомогою списку дійсних чисел (додаткове завдання)**

Реалізувати функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі лабораторної роботи № 6 курсу "Алгоритмізація та програмування" (друга частина), через використання списку дійсних чисел. Кожна пара чисел у масиві має відповідати точці.

1. **Пошук різних слів у реченні (додаткове завдання)**

Увести речення, створити колекцію (SortedSet) різних слів речення та вивести ці слова в алфавітному порядку.

Хід роботи

**Завдання 1. Індивідуальне завдання**

*Код програми*

*Вміст AbstractTopic.java*

|  |
| --- |
| package task\_1;  import java.util.\*;  public abstract class AbstractTopic implements Comparable<AbstractTopic>{  public abstract void setDate(int[] date);  public abstract void setName(String name);  public abstract void setAmount(int amount);    public abstract int[] 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 Arrays.equals(test.getDate(), (getDate())) &&  test.getAmount() == getAmount() &&  test.getName() == getName();    }  @Override  public String toString() { //overloading toString() method  return ("Date: " + getDate()[0] + "." + getDate()[1] + "." + getDate()[2] + " Subject: " + getName() + " Students: " + getAmount());  }  protected void testTopic() {  setDate(new int[]{14,12,2018});  setName("Collections");  setAmount(5);  } } |

*Вміст TopicWithData.java*

|  |
| --- |
| package task\_1;  public class TopicWithData extends AbstractTopic{  private int[] date;  private String subject;  private int students;  public TopicWithData(int[] date, String subject, int students) {  this.date = date;  this.subject = subject;  this.students = students;  }  public TopicWithData() {  // TODO Auto-generated constructor stub  }  @Override  public int[] getDate() {  return date;  }  @Override  public void setDate(int[] date) {  this.date = date;  }  @Override  public String getName() {  return subject;  }  @Override  public void setName(String subject) {  this.subject = subject;  }  @Override  public int getAmount() {  return students;  }  @Override  public void setAmount(int amount) {  this.students = amount;  }    public static void main (String[] args) {  new TopicWithData().testTopic();  } } |

*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 String getName();  public abstract AbstractTopic[] getTopics();  public abstract boolean getExams();  // 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(new int[]{21,9,2018}, "Generics", 8)));  System.out.println(addTopic(new TopicWithData(new int[]{2,10,2018}, "Exceptions", 3)));  System.out.println(addTopic(new TopicWithData(new int[]{16,10,2018}, "GUI", 8)));  System.out.println(addTopic(new TopicWithData(new int[]{30,10,2018}, "Reflection", 7)));  System.out.println(addTopic(new TopicWithData(new int[]{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  } } |

*Вміст SubjectWithArray.java*

|  |
| --- |
| package task\_1; import java.util.\*;  public class SubjectWithArray extends AbstractSubject {  private AbstractTopic[] Topics;  private String name;  private boolean exams;  public SubjectWithArray(AbstractTopic[] topics, String name, boolean exams) {  super();  Topics = topics;  this.name = name;  this.exams = exams;  }  public SubjectWithArray() {  // TODO Auto-generated constructor stub  }   @Override  public AbstractTopic[] getTopics() {  return Topics;  }  @Override  public String getName() {  return name;  }  @Override  public boolean getExams() {  return exams;  }  @Override  public void setTopics(AbstractTopic[] topics) {  this.Topics = topics;  }  @Override  public void setName(String str) {  this.name = str;  }  @Override  public void sortByStudents() {  Arrays.sort(Topics);  }  @Override  public void sortByTopicLength(){  Arrays.sort(Topics, (t1, t2)->Integer.compare(t1.getName().length(),t2.getName().length()));  }  public static void main (String[] args) {  new SubjectWithArray().createSubject().testSubject();  }   } |

*Вміст SubjectWithList.java*

|  |
| --- |
| package task\_1; import java.util.\*;  public class SubjectWithList extends AbstractSubject {  private List<AbstractTopic> Topics = new ArrayList<>();  private String name;  private boolean exams;  public SubjectWithList(List<AbstractTopic> topics, String name, boolean exams) {  super();  Topics = topics;  this.name = name;  this.exams = exams;  }  public SubjectWithList() {  // TODO Auto-generated constructor stub  }  @Override  public AbstractTopic[] getTopics() {  return Topics.toArray(new AbstractTopic[Topics.size()]); // Костыль раз  }  @Override  public String getName() {  return name;  }  @Override  public boolean getExams() {  return exams;  }  @Override  public void setTopics(AbstractTopic[] topics) {  Topics = new ArrayList<>(Arrays.asList(topics)); // Костыль два  }  @Override  public void setName(String str) {  this.name = str;  }  @Override  public void sortByStudents() {  Collections.sort(Topics);  }  @Override  public void sortByTopicLength(){  Collections.sort(Topics, (t1, t2)->Integer.compare(t1.getName().length(),t2.getName().length()));  }  public static void main (String[] args) {  new SubjectWithList().createSubject().testSubject();  }   } |

*Приклад виконання програми*

*Під час запуску SubjectWithArray.java та SubjectWithList.java*

|  |
| --- |
| Adding new practices to OOP true true true true false  Sorting by students amount:  Name: OOP Exams: false Amount of practices: 4 Date: 2.10.2018 Subject: Exceptions Students: 3 Date: 30.10.2018 Subject: Reflection Students: 7 Date: 21.9.2018 Subject: Generics Students: 8 Date: 16.10.2018 Subject: GUI Students: 8  Sorting by the length of a topic name:  Name: OOP Exams: false Amount of practices: 4 Date: 16.10.2018 Subject: GUI Students: 8 Date: 21.9.2018 Subject: Generics Students: 8 Date: 2.10.2018 Subject: Exceptions Students: 3 Date: 30.10.2018 Subject: Reflection Students: 7  Info about lecture containing word Reflection Date: 30.10.2018 Subject: Reflection Students: 7 |

**Завдання 2. Мінімум функції**

*Код програми*

*Вміст test.java*

|  |
| --- |
| package task\_2;  public class test {  static double step = 2;    public static double method(double x) {  return x\*x;  }  public static void main(String[] args) {  System.out.println(MyFunction.find(-5, 5, step, new MyFunction() { // Безымянный класс  @Override  public double f (double x) {  return x\*x;  }  }));    System.out.println(MyFunction.find(-5, 5, step, x -> x\*x)); // Лямбда вражение   System.out.println(MyFunction.find(-5, 5, step, test::method)); // Ссылка на метод  }  } |

*Вміст MyFunction.java*

|  |
| --- |
| package task\_2;  public interface MyFunction{  double f(double x);    static double find (double from, double to, double step, MyFunction func) {  double min=func.f(from);  for (double x = from; x<to; x+=step) {  min = func.f(x)<min?func.f(x):min;  //System.out.println(x);  }  return min;  }   } |

*Приклад виконання програми*

|  |
| --- |
| 1.0  1.0  1.0 |

**Завдання 3. Узагальнений клас**

*Код програми*

|  |
| --- |
| package task\_3;  import java.lang.Object; import java.util.Arrays;  public class MyArray<T> {   private Object[] arr = {};  MyArray(T... arr){  this.arr = arr;  }    MyArray(int size){  arr = new Object[size];  }    public int size() {  return arr.length;  }    public T get(int i) {  return (T)arr[i];  }    public void set (T t, int i) {  arr[i] = t;  }    public void add (T t) {  arr = Arrays.copyOf(arr, arr.length+1);  arr[arr.length-1]=t;  }    public void add(MyArray<T> arr2) { // Добавление массива к массиву  arr=Arrays.copyOf(arr, arr.length+arr2.arr.length);  System.arraycopy(arr2.arr, 0, arr, arr.length-arr2.arr.length, arr2.arr.length);  }    public void remove (T t) {  Object[] temp = new Object[arr.length];  int count=0;  for (Object item: arr) {  if (!t.equals(item)) {  temp[count]=item;  count++;  }  }  arr = Arrays.copyOf(temp, count);  }  @Override  public String toString() {  String str = "";  for (Object item: arr) {  str+=item+" ";  }  return str;  }  public static void main(String[] args) {  MyArray<Integer> array = new MyArray<> (13,3,3);  System.out.println(array); // Исходный массив  array.add(12); // Добавляем элемент  System.out.println(array);  array.remove(3); // Удаляем элемент (все тройки)  System.out.println(array);    MyArray<Integer> array2 = new MyArray<>(7,7,7); // Новый экземпляр массива    array.add(array2); // Добавляем новый массив к старому   System.out.println(array);    }  } |

*Приклад виконання програми*

|  |
| --- |
| 13 3 3  13 3 3 12  13 12  13 12 7 7 7 |

**Завдання 4. Створення бібліотеки узагальнених функцій для роботи з масивами та списками**

*Код програми*

|  |
| --- |
| package task\_4;  import java.util.\*;  public class task\_4 {  public static<T> void swap(T[] arr, int from, int to, int length){ // Меняет местами группу элементов массива  for (int i=0; i<length & to<arr.length; i++, from++, to++) {  T t = arr[from];  arr[from]=arr[to];  arr[to]=t;  }  }  public static<T> void swap(List<T> list, int from, int to, int length) { // Меняет местами группу элементов списка  for (int i=0; i<length & to<list.size(); i++, from++, to++) {  Collections.swap(list, from, to);  }  }  public static<T> void swap2(T[] arr) { // Меняет местами элементы пар массива  for (int i=0; i<arr.length-1; i+=2) {  T t = arr[i+1];  arr[i+1]=arr[i];  arr[i]=t;  }  }  public static<T> void swap2(List<T> list) { // Меняет местами элементы пар листа  for (int i=0; i<list.size()-1; i+=2) {  T t = list.get(i+1);  list.set(i+1, list.get(i));  list.set(i, t);  }  }  public static<T> void replace(T[] arr1, T[] arr2, int i) { // Замена группы элементов в массиве  System.arraycopy(arr2, 0, arr1, i, arr2.length);  }  public static<T> T[] put(T[] arr, T[] arr2, int i) { // Вставка группы элементов в массив  arr = Arrays.copyOf(arr, arr.length+arr2.length);  System.arraycopy(arr, i, arr, i+arr2.length, arr.length-arr2.length-i);  System.arraycopy(arr2, 0, arr, i, arr2.length);  return arr;  }  public static<T> void replace(List<T> list1, List<T> list2, int i) { // Замена группы элементов в листе  put(list1,list2,i);  for (int i1=0; i1<list2.size(); i1++) {  list1.remove(i+list2.size());  }  }  public static<T> void put(List<T> list1, List<T> list2, int i) { // Вставка группы элементов в лист  list1.addAll(i, list2);  }  public static void main(String[] args) {  Integer[] arr = new Integer[] {1,2,3,4,5,6,7,8,9};  System.out.println("Array\n"+Arrays.toString(arr));  swap(arr,1,4,2); // Меняет участки размером 2 по индексам 1, 4  System.out.println(Arrays.toString(arr));  swap2(arr); // Переставляет пары элементов  System.out.println(Arrays.toString(arr));  Integer[] new\_arr = new Integer[] {0, 0, 0};  replace(arr,new\_arr,6); // Заменяет участок одного массива другим  System.out.println(Arrays.toString(arr));  arr = put(arr,new\_arr,2); // Вставляет один массив в другой  System.out.println(Arrays.toString(arr));    List<Integer> list = new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9));  System.out.println("\nList\n"+list);  swap(list,1,4,2); // Меняет участки размером 2 по индексам 1, 4  System.out.println(list);  swap2(list); // Переставляет пары элементов  System.out.println(list);  List<Integer> new\_list = new ArrayList<>(Arrays.asList(0,0,0));  replace(list, new\_list, 6); // Заменяет участок одного списка другим  System.out.println(list);  put(list, new\_list, 2); // Вставляет один список в другой  System.out.println(list);  }  } |

*Приклад виконання програми*

|  |
| --- |
| Array  [1, 2, 3, 4, 5, 6, 7, 8, 9]  [1, 5, 6, 4, 2, 3, 7, 8, 9]  [5, 1, 4, 6, 3, 2, 8, 7, 9]  [5, 1, 4, 6, 3, 2, 0, 0, 0]  [5, 1, 0, 0, 0, 4, 6, 3, 2, 0, 0, 0]  List  [1, 2, 3, 4, 5, 6, 7, 8, 9]  [1, 5, 6, 4, 2, 3, 7, 8, 9]  [5, 1, 4, 6, 3, 2, 8, 7, 9]  [5, 1, 4, 6, 3, 2, 0, 0, 0]  [5, 1, 0, 0, 0, 4, 6, 3, 2, 0, 0, 0] |

**Завдання 5. Реалізація інтерфейсу Comparable**

*Код програми*

|  |
| --- |
| package task\_5;  import java.util.\*;  public class Circle implements Comparable<Circle>{  private double radius;    public Circle (double r) {  this.radius = r;  }  @Override  public int compareTo(Circle c2) {  return Double.compare(radius, c2.radius);  }  public static void sort(List<Circle> list) {  Collections.sort(list);  }  @Override  public String toString () {  return Double.toString(radius);  }    public static void main(String[] args) {  Circle c1 = new Circle(1.2);  Circle c2 = new Circle(3);  Circle c3 = new Circle(6);  Circle c4 = new Circle(3);  List<Circle> list = new ArrayList<>(Arrays.asList(c1,c2,c3,c4));  System.out.println(list);  sort(list);  System.out.println(list);  }  } |

*Приклад виконання програми*

|  |
| --- |
| [1.2, 3.0, 6.0, 3.0]  [1.2, 3.0, 3.0, 6.0] |

**Завдання 6. Реалізація інтерфейсу Comparator**

*Код програми*

|  |
| --- |
| package task\_6;  import java.util.\*;  public class Triangle implements Comparator<Triangle> {  public double a;  public double b;  public double c;    public double getArea() {  return (double)1/4\*Math.sqrt((a+b+c)\*(b+c-a)\*(a+c-b)\*(a+b-c));  }  public Triangle() {}  @Override  public int compare(Triangle tr1, Triangle tr2) {  return Double.compare(tr1.getArea(), tr2.getArea());  }  @Override  public String toString() {  return String.format("%.1f", this.getArea());  }  public static void sort(List<Triangle> list) {  Collections.sort(list, new Triangle());  }  public Triangle(double a, double b, double c) {  this.a = a;  this.b = b;  this.c = c;  }  public static void main(String[] args) {  Triangle tr1 = new Triangle(3,4,5);  Triangle tr2 = new Triangle(5,12,10);  Triangle tr3 = new Triangle(1,1,1);  List<Triangle> list = new ArrayList<>(Arrays.asList(tr1,tr2,tr3));  System.out.println(list);  sort(list);  System.out.println(list);  }  } |

*Приклад виконання програми*

|  |
| --- |
| [6.0, 24.5, 0.4]  [0.4, 6.0, 24.5] |

**Завдання 7. Реалізація масиву точок за допомогою списку дійсних чисел (додаткове завдання)**

*Код програми*

*Вміст AbstractArrayOfPoints.java*

|  |
| --- |
| package task\_7;  public abstract class AbstractArrayOfPoints {  public abstract void setPoint(int i, double x, double y);  public abstract void addPoint(double x, double y);  public abstract void removeLast();  public abstract double getX(int i);  public abstract double getY(int i);  public abstract int count();  public void sortByX() {  for(int i=1;i<count();i++){  for(int j=i; (j>0) && (getX(j-1) > getX(j)); j--){  double x = getX(j);  double y = getY(j);  setPoint(j, getX(j-1), getY(j-1));  setPoint(j-1, x, y);  }  }  }  @Override  public String toString() {  String s = "";  for (int i = 0; i < count(); i++) {  s += "x = " + getX(i) + " \ty = " + getY(i) + "\n";  }  return s + "\n";  }  public void test() {  addPoint(22, 45);  addPoint(4, 11);  addPoint(30, 5.5);  addPoint(-2, 48);  System.out.println(this + "Sorted by X");  sortByX();  System.out.println(this);  }  } |

*Вміст task\_7.java*

|  |
| --- |
| package task\_7;  import java.util.\*;  public class task\_7 extends AbstractArrayOfPoints {  private List<Double> list = new ArrayList<>();    public static void main(String[] args) {  new task\_7().test();  }  @Override  public void setPoint(int i, double x, double y) {  list.set(i\*2, x);  list.set(i\*2+1, y);  }  @Override  public double getX(int i) {  return list.get(i\*2);  }  @Override  public double getY(int i) {  return list.get(i\*2+1);  }  @Override  public int count() {  return list.size()/2;  }  @Override  public void addPoint(double x, double y) {  list.addAll(Arrays.asList(x,y));  }  @Override  public void removeLast() {  // TODO Auto-generated method stub    }  } |

*Приклад виконання програми*

|  |
| --- |
| x = 22.0 y = 45.0  x = 4.0 y = 11.0  x = 30.0 y = 5.5  x = -2.0 y = 48.0  Sorted by X  x = -2.0 y = 48.0  x = 4.0 y = 11.0  x = 22.0 y = 45.0  x = 30.0 y = 5.5 |

**Завдання 8. Пошук різних слів у реченні (додаткове завдання)**

*Код програми*

|  |
| --- |
| package task\_8;  import java.util.\*;  public class task\_8 {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  String str = sc.nextLine();  StringTokenizer st = new StringTokenizer(str);  List<String> list = new ArrayList<>();  while (st.hasMoreTokens()) {  list.add(st.nextToken());  }  Collections.sort(list);  System.out.println(list);  }  } |

*Приклад виконання програми*

|  |
| --- |
| v f a z qwe gf  [a, f, gf, qwe, v, z] |

Висновки

Узагальнення у JAVA виконують схожі за своїм функціоналом на шаблони C++, при тому являючись більш швидкими при обробці, бо працюють тільки з типами-посиланнями, не створюючи окремий клас для кожного реального типу. Колекції представляють собою екземпляри узагальнень, що представляють багатий функціонал для обробки даних. Два базових інтерфейси, що описують методи для роботи с колекціями це Combination та Map.