1 Коллекции и итераторы

1.1.
```
Collection<Person> persons =
    new ArrayList<Person>();
persons.add(new Person("Иванов", 20));
persons.add(new Person("Петров", 21));
persons.add(new Person("Сидорова", 19));
System.out.println(persons);
```

1.2.
```
Iterator<Person> it = persons.iterator();
while (it.hasNext()) {
    Person p = it.next();
    // ...
}
```

1.3.
```
for (Person p : persons) {
    // ...
}
```

1.4.
```
Iterator<Person> it = persons.iterator();
while (it.hasNext()) {
    Person p = it.next();
    if (p.getAge() < 20)
        it.remove();
}
```

2 Списки

2.1.
```
List<Point> points = new LinkedList<Point>();
points.add(new Point(2, -3));
points.add(new Point(0, 0));
points.add(new Point(-3, 8));
```

2.2.
```
points.add(1, new Point(2, 8));
points.set(2, new Point(-1, -1));
points.remove(1);
points.remove(new Point(-3, 8));
```

2.3.
```
ListIterator<Point> it = points.listIterator();
while (it.hasNext()) {
    Point p = it.next();
    if (p.getX() < 0) {
        it.add(new Point(1, 1));
    }
    it.add(new Point(0, 0));
}
```

2.4.
```
ListIterator<Point> it =
    points.listIterator(points.size());
while (it.hasPrevious()) {
    Point p = it.previous();
    if (p.getY() < 0) {
        it.set(new Point(p.getX(), -p.getY()));
    }
}
```

3 Множества

3.1.
```
Set<String> names = new HashSet<String>();
names.add("Вася");
names.add("Маша");
names.add("Саша");
names.add("Вася");
names.add("Даша");
System.out.println(names);
// Даша, Саша, Маша, Вася
```

3.2.
```
boolean hasName = names.contains(newName);
```

3.3.
```
Set<String> names = new TreeSet<String>();
names.add("Вася");
names.add("Маша");
names.add("Саша");
names.add("Вася");
names.add("Даша");
System.out.println(names);
// Вася, Даша, Маша, Саша
```

3.4.
```
Set<Point> points = new TreeSet<Point>();
```

3.5.
```
class Point implements Comparable<Point> {
    // ...
    @Override
    public int compareTo(Point p) {
        if (getX() < p.getX())
            return -1;
        if (getX() > p.getX())
            return 1;
        if (getY() < p.getY())
            return -1;
        if (getY() > p.getY())
            return 1;
        return 0;
    }
    // ...
}
```

3.6.
```
Set<Point> points = new TreeSet<Point>(
        new Comparator<Point>() {
    @Override
    public int compare(Point p1, Point p2) {
        double radDiff = p1.rad() - p2.rad();
        if (radDiff < 0)
            return -1;
        if (radDiff > 0)
            return 1;
        double angleDiff = p1.angle() - p2.angle();
        if (angleDiff < 0)
            return -1;
        if (angleDiff > 0)
            return 1;
        return 0;
    }
});
```

4 Отображения

4.1.
```
Map<String, Person> persons =
    new HashMap<String, Person>();
persons.put("0001", new Person("Иванов", 20));
persons.put("0002", new Person("Петров", 20));
persons.put("0042", new Person("Сидоров", 20));
System.out.println(persons);
```

4.2.
```
boolean hasPerson = persons.containsKey("0042");
```

4.3.
```
Person p = persons.get("0002");
```

4.4.
```
persons.remove("0001");
```

4.5.
```
for (String id : persons.keySet()) {
    // ...
}
```

4.6.
```
for (Person p : persons.values()) {
    // ...
}
```

4.7.
```
for(Map.Entry<String, Person> entry :
        persons.entrySet()) {
    String key = entry.getKey();
    Person p = entry.getValue();
    // ...
}
```

5 Контейнеры и массивы

5.1.
```
Point[] ppp1 = points.toArray(new Point[0]);
System.out.println(Arrays.toString(ppp1));
```

5.2.
```
Point[] ppp2 = new Point[points.size()];
points.toArray(ppp2);
System.out.println(Arrays.toString(ppp2));
```

6 Алгоритмы

6.1.
```
List<String> names = new ArrayList<String>();
names.add("Вася");
names.add("Маша");
names.add("Саша");
names.add("Даша");
Collections.sort(names);
```

6.2.
```
Collections.sort(names, Collections.reverseOrder());
```

6.3.
```
Collections.shuffle(names);
```

6.4.
```
String minName = Collections.min(names);
```

6.5.
```
int idx = Collections.binarySearch(names, "Саша");
```

6.6.
```
List<Point> points = new LinkedList<Point>();
points.add(new Point(2, -3));
points.add(new Point(0, 0));
points.add(new Point(-3, 8));
Collections.sort(points);
```

6.7.
```
Collections.sort(points, pointsComparator);
```

6.8.
```
Collections.sort(names,
    Collections.reverseOrder(pointsComparator));
```

6.9.
```
int idx = Collections.binarySearch(
        points, new Point(-3, 8), pointsComparator);
```