1. Консольный ввод/вывод

1.1.
```
import java.util.Scanner;
import java.io.*;
```

1.2.
```
PrintStream out = System.out;
Scanner in = new Scanner(System.in);
```

1.3.
```
out.println("Как Вас зовут?");
String name = in.nextLine();
out.println("Сколько Вам лет?");
int age = in.nextInt();
out.printf("Вы - %s, Вам %d лет.\n",name,age);
```

1.4.
```
out.println("Введите последовательность:");
int cntPos = 0;
while (true) {
    if (!in.hasNextInt()) {
        out.println("Нужно вводить целые
                                    числа!");
        in.next();
        continue;
    }
    int a = in.nextInt();
    if (a == 0)
        break;
    if (a > 0)
        ++cntPos;
}
out.printf("Вы ввели %d положительных
                        чисел.\n", cntPos);
```

2. Работа с текстовыми файлами

in.txt
```
Василий 24
Петр  23
Анна 24
```

out.txt
```
Имя: Василий; возраст: 24
Имя: Петр; возраст: 23
Имя: Анна; возраст: 24
```

2.1.
```
Scanner inFile = new Scanner(
                        new File(inFileName));
PrintWriter outFile = new PrintWriter(
                                outFileName);
while (inFile.hasNext()) {
    String name = inFile.next();
    int age = inFile.nextInt();
    outFile.printf("Имя: %s; возраст: %d\n",
                                    name, age);
}
inFile.close();
outFile.close();
```

2.2.
```
PrintWriter outFile = new PrintWriter(
      new FileWriter("data/out.txt", true));
```

2.3.
```
Scanner inFile = new Scanner(
              new File(inFileName), "UTF-8");
PrintWriter outFile = new PrintWriter(
                        outFileName, "UTF-8");
```

2.4.
```
PrintWriter outFile = new PrintWriter(
    new OutputStreamWriter(
        new FileOutputStream("data/out2.txt",
                            true), "UTF-8"));
```

2.5.
```
Scanner inFile = new Scanner(
        new BufferedReader(
                new FileReader(inFileName)));
```

2.6.
```
Scanner inFile = new Scanner(
        new BufferedReader(
          new InputStreamReader(
            new FileInputStream(inFileName),
                                    "UTF-8")));
```

3. Двоичные файлы

3.1.
```
Scanner textFile = new Scanner(
                        new File(textFileName));
DataOutputStream binFile =
  new DataOutputStream(
    new BufferedOutputStream(
      new FileOutputStream(binFileName)));
while (textFile.hasNextInt()) {
    int number = textFile.nextInt();
    binFile.writeInt(number);
}
binFile.close();
textFile.close();
```

3.2.
```
DataInputStream binFile =
  new DataInputStream(
    new BufferedInputStream(
      new FileInputStream(binFileName)));
while (binFile.available() > 0) {
    int number = binFile.readInt();
    System.out.println(number);
}
binFile.close();
```

3.3.
```
RandomAccessFile numbers =
    new RandomAccessFile(
                    "data/numbers.dat", "rw");
long length = numbers.length();
final int SIZEOF_INT = 4;
while (numbers.getFilePointer() < length) {
    int number = numbers.readInt();
    number *= 2;
    numbers.seek(
      numbers.getFilePointer() - SIZEOF_INT);
    numbers.writeInt(number);
}
numbers.close();
```

4. Типы исключений

```
java.lang.Object
  |
  +-java.lang.Throwable
     |
     +-java.lang.Error
        |
        +-AssertionError
        +-LinkageError (NoClassDefFoundError)
        +-VirtualMachineError (OutOfMemoryError)
     +-java.lang.Exception
        |
        +-DataFormatException
        +-IOException
        +-RuntimeException
           |
           +-ArithmeticException
           +-ClassCastException
           +-ConcurrentModificationException
           +-EmptyStackException
           +-IllegalArgumentException
           +-IndexOutOfBoundsException
           +-NullPointerException
           +-UnsupportedOperationException
```

5. Обработка исключений

5.1.
```
Scanner inFile = null;
PrintWriter outFile = null;
try {
    try {
        inFile = new Scanner(
                    new File(inFileName));
        outFile = new PrintWriter(
                    outFileName, "UTF-8");
        // ...
    } finally {
        if (inFile != null)
            inFile.close();
        if (outFile != null)
            outFile.close();
    }
} catch (FileNotFoundException e) {
    System.out.println(e.getMessage());
    // e.printStackTrace();
} catch (UnsupportedEncodingException e) {
    System.out.println("Неподдерживаемая
            кодировка: " + e.getMessage());
}
```

5.2.
```
static void printBinaryFile(
                        String binFileName)
    throws FileNotFoundException, IOException {
    DataInputStream binFile = null;
    try {
        binFile = new DataInputStream(
            new FileInputStream(binFileName));
        try {
            while (true) {
              int number = binFile.readInt();
              System.out.println(number);
            }
        } finally {
            if (binFile != null)
                binFile.close();
        }
    } catch (EOFException e) {}
}
```

5.3.
```
String readData(Scanner in)
                        throws EOFException {
    // ...
    while (/* ... */)
    {
      if (!in.hasNext()) {
        if (n < len)
          throw new EOFException();
      }
      // ...
    }
    return s;
}
```

6. Простейшее логирование

6.1.
```
24.10.2007 9:45:18 TestGlobalLogging f
INFO: Method f() is called
24.10.2007 9:45:18 TestGlobalLogging g
INFO: Method g() is called
24.10.2007 9:45:18 TestGlobalLogging g
SEVERE: Exception occured: / by zero
```

6.2.
```
import java.util.logging.*;
public class TestGlobalLogging {
    Logger logger = Logger.getLogger(
                Logger.GLOBAL_LOGGER_NAME);
    void f() {
        logger.info("Method f() is called");
    }
    void g() {
        logger.info("Method g() is called");
        try {
            System.out.println(1 / 0);
        } catch (Exception e) {
            logger.severe("Exception occured:"
                        + e.getMessage());
        }
    }
    public static void main(String[] args) {
        TestGlobalLogging test =
                    new TestGlobalLogging();
        // test.logger.setLevel(Level.OFF);
        test.f();
        test.g();
    }
}
```

6.3.
```
final int size_limit = 1024*1024;// of bytes
final int count_of_files = 5;
final boolean append = true;
FileHandler fh = new
FileHandler("c:\\logs\\mylog.log",
        size_limit, count_of_files, append);
fh.setFormatter(new SimpleFormatter());
test.logger.addHandler(fh);
test.logger.setUseParentHandlers(false);
```

7. Коллекции, итераторы, алгоритмы

7.1.
```
import java.util.*;
```

7.2.
```
Collection<Integer> c =
                    new LinkedList<Integer>();
```

7.3.
```
c.add(3); c.add(5); c.add(2);
```

7.4.
```
System.out.println(c);
```

7.5.
```
int s = 0;
for (int number : c) {
    s += number;
}
```

7.6.
```
Iterator<Integer> iter = c.iterator();
while (iter.hasNext()) {
    int number = iter.next();
    if (number % 2 == 0)
        iter.remove();
}
```

7.7.
```
List<Item> items = new ArrayList<Item>();
// ...
Comparator<Item> itemComparator =
    new Comparator<Item>() {
      public int compare(Item a, Item b) {
        return a.partNumber - b.partNumber;
      }
    };
Collections.sort(items, itemComparator);
```