Input

```
Scanner in = new Scanner(System.in);
  // Can also use new Scanner(new File("input.txt"));

int n = in.nextInt();
double x = in.nextDouble();
String word = in.next();
String line = in.nextLine();

while (in.hasNextDouble())
{
   double x = in.nextDouble();
   Process x
}
```

Output

Does not advance to new line.

```
System.out.print("Enter a value: ");
```

Use + to concatenate values.

```
System.out.println("Volume: " + volume);
```

Field width — Precision

```
System.out.printf("%-10s %10d %10.2f", name, qty, price);
```

Left-justified string — Integer — Floating-point number

```
try (PrintWriter out = new PrintWriter("output.txt"))
{
   Write to out
}
```

Use the print/println/printf methods.

The output is closed at the end of the try-with-resources statement.

Arrays

Element type — Element type — Length — All elements are zero.

```
int[] numbers = new int[5];
int[] squares = { 0, 1, 4, 9, 16 };
int[][] magicSquare =
   {
      { 16,  3,  2, 13},
      {  5, 10, 11,  8},
      {  9,  6,  7, 12},
      {  4, 15, 14,  1}
   };

for (int i = 0; i < numbers.length; i++)
{
   numbers[i] = i * i;
}

for (int element : numbers)
{
   Process element
}

System.out.println(Arrays.toString(numbers));
   // Prints [0, 1, 4, 9, 16]
```

Array Lists

Use wrapper type, Integer, Double, etc., for primitive types. — Initially empty — Element type (optional)

```
ArrayList<String> names = new ArrayList<String>();
```

Add elements to the end

```
names.add("Ann");
names.add("Cindy"); // [Ann, Cindy], names.size() is now 2

names.add(1, "Bob"); // [Ann, Bob, Cindy]
names.remove(2); // [Ann, Bob]
names.set(1, "Bill"); // [Ann, Bill]

String name = names.get(0); // Gets "Ann"
System.out.println(names); // Prints [Ann, Bill]
```

Linked Lists, Sets, and Iterators

```
LinkedList<String> names = new LinkedList<>();
names.add("Bob"); // Adds at end

ListIterator<String> iter = names.listIterator();
iter.add("Ann"); // Adds before current position

String name = iter.next(); // Returns "Ann"
iter.remove(); // Removes "Ann"

Set<String> names = new HashSet<>();
names.add("Ann"); // Adds to set if not present
names.remove("Bob"); // Removes if present

Iterator<String> iter = names.iterator();
while (iter.hasNext())
{
   Process iter.next()
}
```

Maps

Key type — Value type

```
Map<String, Integer> scores = new HashMap<>();

scores.put("Bob", 10);
Integer score = scores.get("Bob");
```

Returns null if key not present

```
for (String key : scores.keySet())
{
   Process key and scores.get(key)
}
```