1. Java is strongly typed language. Every expression and variable has type.
2. **Java types**: Interface, Class (including enum), Array, Primitives.
3. **Primitive Types**: Although java is object-oriented primitive types are not because of performance concerns.
   1. Because of portability reasons java primitives has strict range defined.
   2. **Numbers**: All numbers in java are signed represented in 2’s complement form
      1. Range of number: -2n-1 to 2n-1-1
      2. **Integer**:
         * Byte (8 bit): Useful for working with raw binary data.
         * Short (16 bit):
         * Int (32 bit):
         * Long (64 bit): Used when int type length isn’t sufficient.
      3. **Floatint-point Numbers**: Java uses IEEE-754 for floating point types.
         * For example,
           1. 0.000124 becomes 0.124(M) × 10−3(E)
           2. 237.141 becomes 0.237141 × 10-3
         * Float (32 bit): Represents single-precision value.

S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM

0 1 8 9 31

0 00000000 00000000000000000000000 = 0

1 00000000 00000000000000000000000 = -0

0 11111111 00000000000000000000000 = Infinity

1 11111111 00000000000000000000000 = -Infinity

0 11111111 00000100000000000000000 = NaN

2. Double (64 bit): Floating point literals are by default double and represents double-precision value.

S EEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

0 1 11 12 63

The precision indicates the number of decimal digits that are correct, i.e. without any kind of representation error or approximation. In other words, it indicates how many decimal digits one can safely use.

With that said, it's easy to estimate the number of decimal digits which can be safely used:

single precision: log10(224), which is about 7~8 decimal digits

double precision: log10(253), which is about 15~16 decimal digits

* 1. **Char (16 bit):** Java uses UTF-16 for characters.
     1. Character can used as integer type to perform increment and decrement operarions.
     2. Chars are unsigned integral.
  2. **Boolean:**

1. **Integer literals:**
   1. Octal: 01234567
   2. Hexadecimal:0x0123456789ABCDEF
   3. Binary: 0b101010
   4. Long: 324L
   5. Long integer: 123\_456\_789
      1. These underscores are for clarity and would be ignored. They can’t come at beginning and end of literals.
   6. When int are assigned to byte or long no errors are generated if they are within range.
2. **Floating point literals:**
   1. Float: 2.45f
   2. Hexadecimal: 0x12.2P2
3. **Character literals:** 
   1. Escape sequences: ‘\’’, ‘\n’
   2. Octal: ‘\141’
   3. Hexadecimal: ‘\u0061’
4. Scope: Java has two main scopes: Class and method.
   1. Scope can be at block level.
   2. Object declared in outer scope visible to inner scope.
   3. Variable created and destroyed as scope is entered and exited.
   4. Variable can’t have same name as outer scope.
   5. Uninitialized local variable can’t be used.
5. **Type conversion:** 
   1. Java’s automatic type conversion would take place, if:
      1. Two types are compatible.
      2. Destination type is larger than source type.
   2. For narrowing conversion, we need to cast the variable to destination type.