Section 2: Basic Java Data Types and Statements

1) Fundamental Data Types
   a) Integer
      i) byte, 8 bits, -128 to 127
      ii) short, 16 bits, -32768, 32767
      iii) int, 32 bits, -2147483648, 2147483647
      iv) long, 64 bits, -9223372036854775808, 9223372036854775807
   b) Floating Point
      i) float, 32 bits, +-3.4E+38, +-1.4E-45, 6-7 significant figures
      ii) double, 64 bits, +-1.79E+308, +-4.94E-324, 14-15 significant figures
   c) String - any set of characters in ""
   d) Character - any character in "
   e) Boolean - true, false

2) Statement End
   Java statements end with a semicolon (;)

3) Declarations of Objects of Basic Types
   a) variable identifiers:
      
      \[
      \text{data-type identifier[, identifier[, ...]];}
      \]
      \[
      \text{data-type identifier=expression[, identifier=expression[, ...]];}
      \]

   b) constant identifiers:
      
      \[
      \text{final data-type identifier = expression;}
      \]

4) Declarations of Class Objects
   
   \[
   \text{class identifier;}
   \]
   \[
   \text{identifier = new class(parameter-list);}\]
   
   \[
   \text{class identifier = new class(parameter-list);}\]
5) Comments
   a) from this point on a line rightward:  //
   b) from first mark to second mark:    /* */

6) Assignment
   variable-identifier = expression;

7) Arithmetic Expressions
   operand operator operand [[operator operand] ... ]

8) Arithmetic Operators
   a) integer:  ( ) * / % + -
   b) floating point: ( ) * / + -
   c) string: +

9) Shortcut Operators
   a) ++ - add one to the variable:  ++a a++
   b) -- - subtract one from the variable: --a a--
   c) add the expression to the variable: += // a += b;
   d) subtract the expression from the variable: -= // a -= b;
   e) multiply the variable by the expression: *= // a *= b;
   f) divide the variable by the expression: /= // a /= b;
   g) modulus the variable by the expression: %= // a %= b;

10) Type Conversion (Casting)
    a) (type) variable
    b) (type) (expression)

11) Boolean Expressions
    Abbreviations: e - expression, ro - relational operator, lo - logical operator

    true
    false
    e ro e [lo e ro e [lo e ro e [...]]]

    Note: Parenthesis functions as normal
12) Relational Operators
   a) equal: \( == \)
   b) not equal: \( != \)
   c) less than: \( < \)
   d) greater than: \( > \)
   e) less than or equal to: \( <= \)
   f) greater than or equal to: \( >= \)

13) Logical Operators
   a) Or: \( || \)
   b) And: \( && \)
   c) Not: \( ! \)

14) If Statements
   a) if (boolean) 
      statement
   
   b) if (boolean)
      statement
      else
      statement

15) Switch Statement

   switch(expression)
   {
      case value: statements
      break;
      : 
      default: statements
   };

16) While Loop

   while (boolean-expression)
   statement
17) **Do-While Loop**

```java
do {
    statements
} while (boolean-expression);
```

18) **For Loop**

```java
for (init-statement; boolean-expression; counter)
    statements
```

19) **Arrays**

a) declaring:

```java
type-or-class [] identifier;
identifier = new type-or-class[size];

type-or-class [] identifier = new type-or-class[size];

type-or-class [] identifier = {data, data, ..., data};

type-or-class [][] identifier;
identifier = new type-or-class[numRows][numCols];

type-or-class identifier = new type-or-class[numRows][numCols];

type-or-class [][] identifier = {{data, data, ..., data},
                                    {data, data, ..., data},
                                    ...{data, data, ..., data}};
```

b) indices: 0 - (size-1)
c) length: `arrayIdentifier.length` or `arrayIdentifier[row].length`
d) arrays are pass by reference (objects of a class are pass by reference)
e) accessing: `arrayIdentifier[i]`  
    `arrayIdentifier[i][j]`  
    `arrayIdentifier[i][j]...[n]`