Selection

General form of `if` Statements

```
if (condition) {
   statement1;
}
else {
   statement2;
}
```

Each statement may be a single statement or a compound statement enclosed in curly braces. The condition is any expression that returns a **boolean** value.

Nested if Statement

```
if (condition) {
   if (condition) {
      statement;
   }
   else {
      statements;
   }
else {
statement;
}
```

An `else` statement always refers to the nearest `if` statement.

General form of `if-else-if` Statements

```
if (condition) {
   statement;
else if (condition) {
   statement;
else if (condition) {
}
   statement;
}
}
⋮
else {
   statement;
}
```

The `if` statements are executed from the top down. As soon as one of the conditions controlling the `if` is **true**, the statement associated with that `if` is executed, and the rest of the code is bypassed. If none of the conditions is true, then the final `else` statement will be executed. The final `else` statement is the default condition; that is, if all other conditional test fail, then the last `else` statement is performed. If there is no final `else` and all conditions are **false**, then no action will take place.

General form of `switch` statement

```
switch (expression) {
case value1:
        statement;
        break;
case value2:
⋮
case valueN:
        statement;
}
```

The expression must be of type `byte, short, int, char, String`, or an enumeration. Each value specified in the `case` statements must be a unique constant expression. Duplicate `case` values are not allowed. The type of each value must be compatible with the tyoe of expression.

A constant expression is an expression that yields a primitive type or a String, and whose value can be evaluated at compile time to a literal.

The `break` statement is used inside the `switch` statement to terminate a statement sequence. When a `break` statement is encountered, execution branches to the first line of code that follows the entire switch statement.

The `break` statement is optional. If you omit the `break`, execution will continue on into the next `case`.

Important features of the `switch` statement

- The `switch` differes from the `if` in the fact that `switch` can only test for equality, whereas `if` can evaluate any type of Boolean expression. The `switch` looks only for a match betwen the value of the expression and one of its `case` constants.
- No two `case` constants in the same `switch` can have identical values.
- A `switch` statement is usually more efficient than a *set* of nested `if`s.