Solutions to Quick Check Questions

5

Selection Statements

5.1 The if Statement

1. Identify the invalid if statements:

a.
```
if ( a < b ) then
   x = y;
else
   x = z;
```

b.
```
if ( a < b )
   else x = y;
```

c.
```
if ( a < b )
   x = y;
   else {
   x = z;
   };
```

d.
```
if ( a < b ) {
   x = y; } else
   x = z;
```

a. Invalid. The word then *is not a reserved word in Java.*

b. Invalid. No statement is the then part. If you place the braces as

```
if (a < b) { }
else x = y;
```

then it is valid.

c. Valid. Semicolon after the right brace is never necessary, but this statement actually compiles without generating any error message. The Java compiler actually reads this statment as an if statement followed by an empty statement. The semicolon is terminating the empty statement.

d. Valid (even though it is not formatted properly).

2. Are the following two if statements equivalent?

```
/*A*/  if ( x < y )
           System.out.println("Hello");
       else
           System.out.println("Bye");

/*B*/  if ( x > y )
           System.out.println("Bye");
       else
           System.out.println("Hello");
       }
```

No. When the variables x and y have the same value, the A code prints out Bye, *while the B code prnts out* Hello.

5.2 Nested-if Statements

1. Rewrite the following nested-if statements without using any nesting:

```
a.     if ( a < c )
           if ( b < c )
               x = y;
           else
               x = z;
       else
           x = z;

b.     if ( a == b )
           x = y;
       else
           if ( a > b )
               x = y;
           else
               x = z;

c.      if ( a < b  )
               if ( a >= b )
                   x = z;
```

```
   else
      x = y;
else
   x = z;
```

Answers:

. a.
```
if ( a < c && b < c)
      x = y;
else
      x = z;
```

b.
```
if ( a => b )
   x = y;
else
   x = z;
```

c.
```
 if ( a < b  )
   x = y;
else
   x = z;
```

2. Format the following if statements with indentation.

a.
```
if ( a < b  ) if ( c > d ) x = y;
else x = z;
```

b.
```
if ( a < b  ) { if ( c > d ) x = y; }
else x = z;
```

c.
```
if ( a < b ) x = y; if ( a < c ) x = z;
else if ( c < d ) z = y;
```

Answers:

a.
```
if ( a < b  )
   if ( c > d )
      x = y;
   else
      x = z;
```

b.
```
if ( a < b  ) {
   if ( c > d )
      x = y;
   }
else
```

```
            x = z;

c.      if ( a < b )
            x = y;
        if ( a < c )
            x = z;
        else if (c < d)
            z = y;
```

5.3 Boolean Expressions and Variables

1. Evaluate the following boolean expressions. Assume x, y, and z have some numerical values.

```
a.      4 < 5 || 6 == 6
b.      2 < 4 && (false || 5 <= 4)
c.      x <= y && !(z != z) || x > y
d.      x < y || z < y && y <= z
```

a. true

b. false

c. true. The expression !(z != z) *is always true. If x <= y then the x <= y && !(z!=z) is true so the whole expression is true. If x > y, then the whole expression is also true.*

d. result of x < y. The expression z < y && y <=z is always false so the whole expression is true if x < y is true and false if x < y is false.

2. Identify errors in the following boolean expressions and assignments.

```
a.      boolean done;
        done = x = y;

b.      2 < 4 && (3 < 5) + 1 == 3

c.      boolean quit;
        quit = true;
        quit == ( 34 == 20 ) && quit;
```

a. Assuming that x and y are some numerical data type, the second statement should be

```
done = x == y;
```

b. The result of (3 < 5) is a boolean value so adding 1 to it is invalid.

c. The third statement should be

```
quit = ( 34 == 20 ) && quit;
```

5.4 Comparing Objects

1. Determine the output of the following code:

```
String str1 = "Java";
String str2 = "Java";

boolean result1 = str1 == str2;
boolean result2 = str1.equals(str2);

System.out.println(result1);
System.out.println(result2);
```

Answer:

```
true
true
```

2. Determine the output of the following code:

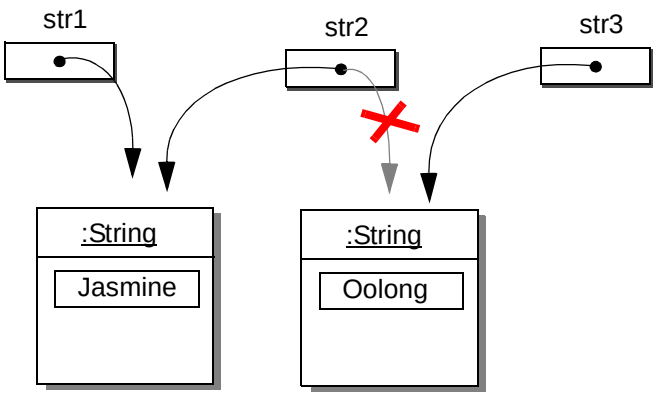
```
String str1 = new String("latte");
String str2 = new String("LATTE");

boolean result1 = str1 == str2;
boolean result2 = str1.equals(str2);

System.out.println(result1);
System.out.println(result2);
```

Answer:

```
false
false
```

3. Show the state of memory after the following statements are executed:

```
String str1, str2, str3;
str1 = "Jasmine";
str2 = "Oolong";
str3 = str2;
str2 = str1;
```



5.5 The switch Statement

1. What's wrong with the following switch statement?

```
switch ( N ) {
   case  0:
   case  1:  x = 11;
             break;
   default:  System.out.println("Switch Error");
             break;
   case  2:  x = 22;
             break;
   case  1:  x = 33;
             break;
}
```

The case label 1 *is repeated twice. Duplicating the case label is invalid. Note: Although placing the* default *label at the end of the* switch *statement is preferable, it is not a requirement.*

2. What's wrong with the following switch statement?

```
switch ( ranking ) {
   case  >4.55:  pay = pay * 0.20;
                 break;

   case  =4.55:  pay = pay * 0.15;
                 break;

   default:      pay = pay * 0.05;
                 break;
}
```

The case labels must be an integral value and equality is the only comparison allowed. The labels >4.55 *and* =4.55 *are therefore invalid.*

5.6 Drawing Graphics

No Quick Check Questions.

5.7 Enmerated Constants

1. Define an enumerated type Day that includes the constants SUNDAY through SATURDAY.

   Answer:

```
enum Day {SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
          THURSDAY, FRIDAY, SATURDAY}
```

2. What is the method that returns an enumerated constant, given the matching String value?

   Answer:

```
valueOf
```

3. Detect the error(s) in the following code:

```
enum Fruit {APPLE, ORANGE, BANANA}
Fruit f1, f2;
int f3;
f1 = 1;      <---Invalid: type incompatible

f2 = ORANGE;  <--Invalid: need enum type as in Fruit.ORANGE

f3 = f1;      <---Invalid: type incompatible

f1 = "BANANA"; <---Invalid: type incompatible
```

5.8 Sample Program: Drawing Shapes

No Quick Check Questions.