Review Questions

The answers to the chapter review questions can be found in the Appendix.

1. Which statements about the final modifier are correct? (Choose all that apply.)

   A. Instance and static variables can be marked final.

   B. A variable is effectively final only if it is marked final.

   C. An object that is marked final cannot be modified.

   D. Local variables cannot be declared with type var and the final modifier.

   E. A primitive that is marked final cannot be modified.

2. Which of the following can fill in the blank in this code to make it compile? (Choose all that apply.)

```
public class Ant {
  _______ void method() {}
}
```

   A. default

   B. final

   C. private

   D. Public

   E. String

   F. zzz:

3. Which of the following methods compile? (Choose all that apply.)

   A. final static void rain() {}

   B. public final int void snow() {}

   C. private void int hail() {}

   D. static final void sleet() {}

   E. void final ice() {}

   F. void public slush() {}

4. Which of the following can fill in the blank and allow the code to compile? (Choose all that apply.)

```
final _____ song = 6;
```

A. int

B. Integer

C. long

D. Long

E. double

F. Double

5. Which of the following methods compile? (Choose all that apply.)

A. public void january() { return; }

B. public int february() { return null;}

C. public void march() {}

D. public int april() { return 9;}

E. public int may() { return 9.0;}

F. public int june() { return;}

6. Which of the following methods compile? (Choose all that apply.)

A. public void violin(int... nums) {}

B. public void viola(String values, int... nums) {}

C. public void cello(int... nums, String values) {}

D. public void bass(String... values, int... nums) {}

E. public void flute(String[] values, ...int nums) {}

F. public void oboe(String[] values, int[] nums) {}

7. Given the following method, which of the method calls return 2? (Choose all that apply.)

```
public int juggle(boolean b, boolean... b2) {
  return b2.length;
}
```

A. juggle();

B. juggle(true);

C. juggle(true, true);

D. juggle(true, true, true);

E. juggle(true, {true, true});

F. juggle(true, new boolean[2]);

8. Which of the following statements is correct?

A. Package access is more lenient than protected access.

B. A public class that has private fields and package methods is not visible to classes outside the package.

C. You can use access modifiers so only some of the classes in a package see a particular package class.

D. You can use access modifiers to allow access to all methods and not any instance variables.

E. You can use access modifiers to restrict access to all classes that begin with the word Test.

9. Given the following class definitions, which lines in the main() method generate a compiler error? (Choose all that apply.)

```
// Classroom.java
package my.school;
public class Classroom {
  private int roomNumber;
  protected static String teacherName;
  static int globalKey = 54321;
  public static int floor = 3;
  Classroom(int r, String t) {
    roomNumber = r;
    teacherName = t; } }

// School.java
1: package my.city;
2: import my.school.*;
3: public class School {
4:   public static void main(String[] args) {
5:     System.out.println(Classroom.globalKey);
6:     Classroom room = new Classroom(101, "Mrs. Anderson");
7:     System.out.println(room.roomNumber);
8:     System.out.println(Classroom.floor);
9:     System.out.println(Classroom.teacherName); } }
```

A. None: the code compiles fine.

B. Line 5

C. Line 6

D. Line 7

E. Line 8

F. Line 9

10. What is the output of executing the Chimp program?

// Rope.java

```
1: package rope;
2: public class Rope {
3:   public static int LENGTH = 5;
4:   static {
5:     LENGTH = 10;
6:   }
7:   public static void swing() {
8:     System.out.print("swing ");
9:   }}
```

// Chimp.java

```
1: import rope.*;
2: import static rope.Rope.*;
3: public class Chimp {
4:   public static void main(String[] args) {
5:     Rope.swing();
6:     new Rope().swing();
7:     System.out.println(LENGTH);
8:   }}
```

A. swing swing 5

B. swing swing 10

C. Compiler error on line 2 of Chimp

D. Compiler error on line 5 of Chimp

E. Compiler error on line 6 of Chimp

F. Compiler error on line 7 of Chimp

11. Which statements are true of the following code? (Choose all that apply.)

```
1:  public class Rope {
2:    public static void swing() {
3:      System.out.print("swing");
4:    }
5:    public void climb() {
6:      System.out.println("climb");
7:    }
8:    public static void play() {
9:      swing();
10:     climb();
```

```
11:   }
12:   public static void main(String[] args) {
13:     Rope rope = new Rope();
14:     rope.play();
15:     Rope rope2 = null;
16:     System.out.print("-");
17:     rope2.play();
18:   } }
```

A. The code compiles as is.

B. There is exactly one compiler error in the code.

C. There are exactly two compiler errors in the code.

D. If the line(s) with compiler errors are removed, the output is swing-climb.

E. If the line(s) with compiler errors are removed, the output is swing-swing.

F. If the line(s) with compile errors are removed, the code throws a NullPointerException.

12. How many variables in the following method are effectively final?

```
10: public void feed() {
11:   int monkey = 0;
12:   if(monkey > 0) {
13:     var giraffe = monkey++;
14:     String name;
15:     name = "geoffrey";
16:   }
17:   String name = "milly";
18:   var food = 10;
19:   while(monkey <= 10) {
20:     food = 0;
21:   }
22:   name = null;
23: }
```

A. 1

B. 2

C. 3

D. 4

E. 5

F. None of the above. The code does not compile.

13. What is the output of the following code?

```
// RopeSwing.java
import rope.*;
import static rope.Rope.*;
public class RopeSwing {
  private static Rope rope1 = new Rope();
  private static Rope rope2 = new Rope();
  {
    System.out.println(rope1.length);
  }
  public static void main(String[] args) {
    rope1.length = 2;
    rope2.length = 8;
    System.out.println(rope1.length);
  }
}

// Rope.java
package rope;
public class Rope {
  public static int length = 0;
}
```

A. 02

B. 08

C. 2

D. 8

E. The code does not compile.

F. An exception is thrown.

14. How many lines in the following code have compiler errors?

```
1:  public class RopeSwing {
2:    private static final String leftRope;
3:    private static final String rightRope;
4:    private static final String bench;
5:    private static final String name = "name";
6:    static {
7:      leftRope = "left";
8:      rightRope = "right";
9:    }
10:   static {
11:     name = "name";
12:     rightRope = "right";
```

```
13:   }
14:   public static void main(String[] args) {
15:     bench = "bench";
16:   }
17: }
```

A. 0

B. 1

C. 2

D. 3

E. 4

F. 5

15. Which of the following can replace line 2 to make this code compile? (Choose all that apply.)

```
1: import java.util.*;
2: // INSERT CODE HERE
3: public class Imports {
4:   public void method(ArrayList<String> list) {
5:     sort(list);
6:   }
7: }
```

A. import static java.util.Collections;

B. import static java.util.Collections.*;

C. import static java.util.Collections.sort(ArrayList<String>);

D. static import java.util.Collections;

E. static import java.util.Collections.*;

F. static import java.util.Collections.sort(ArrayList<String>);

16. What is the result of the following statements?

```
1: public class Test {
2:   public void print(byte x) {
3:     System.out.print("byte-");
4:   }
5:   public void print(int x) {
6:     System.out.print("int-");
7:   }
8:   public void print(float x) {
9:     System.out.print("float-");
10:  }
```

```
11:   public void print(Object x) {
12:     System.out.print("Object-");
13:   }
14:   public static void main(String[] args) {
15:     Test t = new Test();
16:     short s = 123;
17:     t.print(s);
18:     t.print(true);
19:     t.print(6.789);
20:   }
21: }
```

A. byte-float-Object-

B. int-float-Object-

C. byte-Object-float-

D. int-Object-float-

E. int-Object-Object-

F. byte-Object-Object-

17. What is the result of the following program?

```
1:  public class Squares {
2:    public static long square(int x) {
3:      var y = x * (long) x;
4:      x = -1;
5:      return y;
6:    }
7:    public static void main(String[] args) {
8:      var value = 9;
9:      var result = square(value);
10:      System.out.println(value);
11:   } }
```

A. -1

B. 9

C. 81

D. Compiler error on line 9

E. Compiler error on a different line

18. Which of the following are output by the following code? (Choose all that apply.)

```
public class StringBuilders {
  public static StringBuilder work(StringBuilder a,
    StringBuilder b) {
    a = new StringBuilder("a");
    b.append("b");
    return a;
  }
  public static void main(String[] args) {
    var s1 = new StringBuilder("s1");
    var s2 = new StringBuilder("s2");
    var s3 = work(s1, s2);
    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
    System.out.println("s3 = " + s3);
  }
}
```

A. s1 = a

B. s1 = s1

C. s2 = s2

D. s2 = s2b

E. s3 = a

F. The code does not compile.

19. Which of the following will compile when independently inserted in the following code? (Choose all that apply.)

```
1:  public class Order3 {
2:    final String value1 = "red";
3:    static String value2 = "blue";
4:    String value3 = "yellow";
5:    {
6:      // CODE SNIPPET 1
7:    }
8:    static {
9:      // CODE SNIPPET 2
10:   }}
```

A. Insert at line 6: value1 = "green";

B. Insert at line 6: value2 = "purple";

C. Insert at line 6: value3 = "orange";

D. Insert at line 9: value1 = "magenta";

E. Insert at line 9: value2 = "cyan";

F. Insert at line 9: value3 = "turquoise";

20. Which of the following are true about the following code? (Choose all that apply.)

```
public class Run {
  static void execute() {
    System.out.print("1-");
  }
  static void execute(int num) {
    System.out.print("2-");
  }
  static void execute(Integer num) {
    System.out.print("3-");
  }
  static void execute(Object num) {
    System.out.print("4-");
  }
  static void execute(int... nums) {
    System.out.print("5-");
  }
  public static void main(String[] args) {
    Run.execute(100);
    Run.execute(100L);
  }
}
```

A. The code prints out 2-4-.

B. The code prints out 3-4-.

C. The code prints out 4-2-.

D. The code prints out 4-4-.

E. The code prints 3-4- if you remove the method static void execute(int num).

F. The code prints 4-4- if you remove the method static void execute(int num).

21. Which method signatures are valid overloads of the following method signature? (Choose all that apply.)

```
public void moo(int m, int... n)
```

A. public void moo(int a, int... b)

B. public int moo(char ch)

C. public void moooo(int... z)

D. private void moo(int... x)

E. public void moooo(int y)

F. public void moo(int... c, int d)

G. public void moo(int... i, int j...)

Soluciones:

1. A, E. Instance and static variables can be marked final, making option A correct. Effectively final means a local variable is not marked final but whose value does not change after it is set, making option B incorrect. Option C is incorrect, as final refers only to the reference to an object, not its contents. Option D is incorrect, as var and final can be used together. Finally, option E is correct: once a primitive is marked final, it cannot be modified.

2. B, C. The keyword void is a return type. Only the access modifier or optional specifiers are allowed before the return type. Option C is correct, creating a method with private access. Option B is also correct, creating a method with package access and the optional specifier final. Since package access does not use a modifier, we get to jump right to final. Option A is incorrect because package access omits the access modifier rather than specifying default. Option D is incorrect because Java is case sensitive. It would have been correct if public were the choice. Option E is incorrect because the method already has a void return type. Option F is incorrect because labels are not allowed for methods.

3. A, D. Options A and D are correct because the optional specifiers are allowed in any order. Options B and C are incorrect because they each have two return types. Options E and F are incorrect because the return type is before the optional specifier and access modifier, respectively.

4. A, B, C, E. The value 6 can be implicitly promoted to any of the primitive types, making options A, C, and E correct. It can also be autoboxed to Integer, making option B correct. It cannot be both promoted and autoboxed, making options D and F incorrect.

5. A, C, D. Options A and C are correct because a void method is optionally allowed to have a return statement as long as it doesn't try to return a value. Option B does not compile because null requires a reference object as the return type. Since int is primitive, it is not a reference object. Option D is correct because it returns an int value. Option E does not compile because it tries to return a double when the return type is int. Since a double cannot be assigned to an int, it cannot be returned as one either. Option F does not compile because no value is actually returned.

6. A, B, F. Options A and B are correct because the single varargs parameter is the last parameter declared. Option F is correct because it doesn't use any varargs parameters. Option C is incorrect because the varargs parameter is not last. Option D is incorrect because two varargs parameters are not allowed in the same method. Option E is incorrect because the ... for a varargs must be after the type, not before it.

7. D, F. Option D passes the initial parameter plus two more to turn into a varargs array of size 2. Option F passes the initial parameter plus an array of size 2. Option A does not compile because it does not pass the initial parameter. Option E does not compile because it does not declare an array properly. It should be new boolean[] {true, true}. Option B creates a varargs array of size 0, and option C creates a varargs array of size 1.

8. D. Option D is correct. A common practice is to set all fields to be private and all methods to be public. Option A is incorrect because protected access allows everything that package access allows and additionally allows subclasses access. Option B is incorrect because the class is public. This means that other classes can see the class. However, they cannot call any of the methods or read any of the fields. It is essentially a useless class. Option C is incorrect because package access applies to the whole package. Option E is incorrect because Java has no such wildcard access capability.

9. B, C, D, F. The two classes are in different packages, which means private access and package access will not compile. This causes compiler errors on lines 5, 6, and 7, making options B, C, and D correct answers. Additionally, protected access will not compile since School does not inherit from Classroom. This causes the compiler error on line 9, making option F a correct answer as well.

10. B. Rope runs line 3, setting LENGTH to 5, and then immediately after that runs the static initializer, which sets it to 10. Line 5 in the Chimp class calls the static method normally and prints swing and a space. Line 6 also calls the static method. Java allows calling a static method through an instance variable, although it is not recommended. Line 7 uses the static import on line 2 to reference LENGTH. For these reasons, option B is correct.

11. B, E. Line 10 does not compile because static methods are not allowed to call instance methods. Even though we are calling play() as if it were an instance method and an instance exists, Java knows play() is really a static method and treats it as such. Since this is the only line that does not compile, option B is correct. If line 10 is removed, the code prints swing-swing, making option E correct. It does not throw a NullPointerException on line 17 because play() is a static method. Java looks at the type of the reference for rope2 and translates the call to Rope.play().

12. B. The test for effectively final is if the final modifier can be added to the local variable and the code still compiles. The monkey variable declared on line 11 is not effectively final because it is modified on line 13. The giraffe and name variables declared on lines 13 and 14, respectively, are effectively final and not modified after they are set. The name variable declared on line 17 is not effectively final since it is modified on line 22. Finally, the food variable on line 18 is not effectively final since it is modified on line 20. Since there are two effectively final variables, option B is correct.

13. D. There are two details to notice in this code. First, note that RopeSwing has an instance initializer and not a static initializer. Since RopeSwing is never constructed, the instance initializer does not run. The other detail is that length is static. Changes from any object update this common static variable. The code prints 8, making option D correct.

14. E. If a variable is static final, it must be set exactly once, and it must be in the declaration line or in a static initialization block. Line 4 doesn't compile because bench is not set in either of these locations. Line 15 doesn't compile because final variables are not allowed to be set after that point. Line 11 doesn't compile because name is set twice: once in the declaration and again in the static block. Line 12 doesn't compile because rightRope is set twice as well. Both are in static initialization blocks. Since four lines do not compile, option E is correct.

15. B. The two valid ways to do this are import static java.util.Collections.*; and import static java.util.Collections.sort;. Option A is incorrect because you can do a static import only on static members. Classes such as Collections require a regular import. Option C is nonsense as method parameters have no business in an import. Options D, E, and F try to trick you into reversing the syntax of import static.

16. E. The argument on line 17 is a short. It can be promoted to an int, so print() on line 5 is invoked. The argument on line 18 is a boolean. It can be autoboxed to a Boolean, so print() on line 11 is invoked. The argument on line 19 is a double. It can be autoboxed to a Double, so print() on line 11 is invoked. Therefore, the output is int-Object-Object-, and the correct answer is option E.

17. B. Since Java is pass-by-value and the variable on line 8 never gets reassigned, it stays as 9. In the method square, x starts as 9. The y value becomes 81, and then x gets set to –1. Line 9 does set result to 81. However, we are printing out value, and that is still 9, making option B correct.

18. B, D, E. Since Java is pass-by-value, assigning a new object to a does not change the caller. Calling append() does affect the caller because both the method parameter and the caller have a reference to the same object. Finally, returning a value does pass the reference to the caller for assignment to s3. For these reasons, options B, D, and E are correct.

19. B, C, E. The variable value1 is a final instance variable. It can be set only once: in the variable declaration, an instance initializer, or a constructor. Option A does not compile because the final variable was already set in the declaration. The variable value2 is a static variable. Both instance and static initializers are able to access static variables, making options B and E correct. The variable value3 is an instance variable. Options D and F do not compile because a static initializer does not have access to instance variables.

20. A, E. The 100 parameter is an int and so calls the matching int method, making option A correct. When this method is removed, Java looks for the next most specific constructor. Java prefers autoboxing to varargs, so

it chooses the Integer constructor. The 100L parameter is a long. Since it can't be converted into a smaller type, it is autoboxed into a Long, and then the method for Object is called, making option E correct.

21. B, D. Option A is incorrect because it has the same parameter list of types and therefore the same signature as the original method. Options B and D are valid method overloads because the types of parameters in the list change. When overloading methods, the return type and access modifiers do not need to be the same. Options C and E are incorrect because the method name is different. Options F and G do not compile. There can be at most one varargs parameter, and it must be the last element in the parameter list.