1. 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:

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

A. final static void method4() { }

B. public final int void method() { }

C. private void int method() { }

D. static final void method3() { }

E. void final method() {}

F. void public method() { }

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

A. public void methodA() { return;}

B. public void methodB() { return null;}

C. public void methodD() {}

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

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

F. public int methodF() { return;}

G. public int methodG() { return null;}

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

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

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

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

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

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

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

G. public void moreG(String[] values, int[] nums) {}

Review Questions 219

c04.indd 1½ 4/2014 Page 219

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

public int howMany(boolean b, boolean... b2) {

return b2.length;

}

A. howMany();

B. howMany(true);

C. howMany(true, true);

D. howMany(true, true, true);

E. howMany(true, {true});

F. howMany(true, {true, true});

G. howMany(true, new boolean[2]);

6. Which of the following are true? (Choose all that apply)

A. Package private access is more lenient than protected access.

B. A public class that has private fields and package private 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 private class.

D. You can use access modifiers to allow read access to all methods, but not any instance

variables.

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

word Test.

7. Given the following my.school.ClassRoom and my.city.School class definitions, which

line numbers in main() generate a compiler error? (Choose all that apply)

1: package my.school;

2: public class Classroom {

3: private int roomNumber;

4: protected String teacherName;

5: static int globalKey = 54321;

6: public int floor = 3;

7: Classroom(int r, String t) {

8: roomNumber = r;

9: teacherName = t; } }

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");

220 Chapter 4 ■ Methods and Encapsulation

c04.indd 1½ 4/2014 Page 220

7: System.out.println(room.roomNumber);

8: System.out.println(room.floor);

9: System.out.println(room.teacherName); } }

A. None, the code compiles fine.

B. Line 5

C. Line 6

D. Line 7

E. Line 8

F. Line 9

8. Which of the following are true? (Choose all that apply)

A. Encapsulation uses package private instance variables.

B. Encapsulation uses private instance variables.

C. Encapsulation allows setters.

D. Immutability uses package private instance variables.

E. Immutability uses private instance variables.

F. Immutability allows setters.

9. Which are methods using JavaBeans naming conventions for accessors and mutators?

(Choose all that apply)

A. public boolean getCanSwim() { return canSwim;}

B. public boolean canSwim() { return numberWings;}

C. public int getNumWings() { return numberWings;}

D. public int numWings() { return numberWings;}

E. public void setCanSwim(boolean b) { canSwim = b;}

10. What is the output of the following code?

1: package rope;

2: public class Rope {

3: public static int LENGTH = 5;

4: static {

5: LENGTH = 10;

6: }

Review Questions 221

c04.indd 1½ 4/2014 Page 221

7: public static void swing() {

8: System.out.print("swing ");

9: }

10: }

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: }

9: }

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 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: rope2.play();

17: }

18: }

222 Chapter 4 ■ Methods and Encapsulation

c04.indd 1½ 4/2014 Page 222

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 lines with compiler errors are removed, the output is climb climb.

E. If the lines with compiler errors are removed, the output is swing swing.

F. If the lines with compile errors are removed, the code throws a NullPointerException.

12. What is the output of the following code?

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);

}

}

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.

13. How many compiler errors are in the following code?

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";

Review Questions 223

c04.indd 1½ 4/2014 Page 223

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

14. 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>);

15. 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");

224 Chapter 4 ■ Methods and Encapsulation

c04.indd 1½ 4/2014 Page 224

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. bytefloatObject

B. intfloatObject

C. byteObjectfloat

D. intObjectfloat

E. intObjectObject

F. byteObjectObject

16. What is the result of the following program?

1: public class Squares {

2: public static long square(int x) {

3: long y = x \* (long) x;

4: x = -1;

5: return y;

6: }

7: public static void main(String[] args) {

8: int value = 9;

9: long 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.

Review Questions 225

c04.indd 1½ 4/2014 Page 225

17. 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) {

StringBuilder s1 = new StringBuilder("s1");

StringBuilder s2 = new StringBuilder("s2");

StringBuilder 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. s3 = null

G. The code does not compile.

18. Which of the following are true? (Choose 2)

A. this() can be called from anywhere in a constructor.

B. this() can be called from any instance method in the class.

C. this.variableName can be called from any instance method in the class.

D. this.variableName can be called from any static method in the class.

E. You must include a default constructor in the code if the compiler does not include one.

F. You can call the default constructor written by the compiler using this().

G. You can access a private constructor with the main() method.

19. Which of these classes compile and use a default constructor? (Choose all that apply)

A. public class Bird { }

B. public class Bird { public bird() {} }

C. public class Bird { public bird(String name) {} }

D. public class Bird { public Bird() {} }

226 Chapter 4 ■ Methods and Encapsulation

c04.indd 1½ 4/2014 Page 226

E. public class Bird { Bird(String name) {} }

F. public class Bird { private Bird(int age) {} }

G. public class Bird { void Bird() { }

20. Which code can be inserted to have the code print 2?

public class BirdSeed {

private int numberBags;

boolean call;

public BirdSeed() {

// LINE 1

call = false;

// LINE 2

}

public BirdSeed(int numberBags) {

this.numberBags = numberBags;

}

public static void main(String[] args) {

BirdSeed seed = new BirdSeed();

System.out.println(seed.numberBags);

} }

A. Replace line 1 with BirdSeed(2);

B. Replace line 2 with BirdSeed(2);

C. Replace line 1 with new BirdSeed(2);

D. Replace line 2 with new BirdSeed(2);

E. Replace line 1 with this(2);

F. Replace line 2 with this(2);

21. Which of the following complete the constructor so that this code prints out 50? (Choose

all that apply)

public class Cheetah {

int numSpots;

public Cheetah(int numSpots) {

// INSERT CODE HERE

}

public static void main(String[] args) {

System.out.println(new Cheetah(50).numSpots);

}

}

Review Questions 227

c04.indd 1½ 4/2014 Page 227

A. numSpots = numSpots;

B. numSpots = this.numSpots;

C. this.numSpots = numSpots;

D. numSpots = super.numSpots;

E. super.numSpots = numSpots;

F. None of the above.

22. What is the result of the following?

1: public class Order {

2: static String result = "";

3: { result += "c"; }

4: static

5: { result += "u"; }

6: { result += "r"; }

7: }

1: public class OrderDriver {

2: public static void main(String[] args) {

3: System.out.print(Order.result + " ");

4: System.out.print(Order.result + " ");

5: new Order();

6: new Order();

7: System.out.print(Order.result + " ");

8: }

9: }

A. curur

B. ucrcr

C. u ucrcr

D. u u curcur

E. u u ucrcr

F. ur ur urc

G. The code does not compile.

23. What is the result of the following?

1: public class Order {

2: String value = "t";

3: { value += "a"; }

4: { value += "c"; }

5: public Order() {

228 Chapter 4 ■ Methods and Encapsulation

c04.indd 1½ 4/2014 Page 228

6: value += "b";

7: }

8: public Order(String s) {

9: value += s;

10: }

11: public static void main(String[] args) {

12: Order order = new Order("f");

13: order = new Order();

14: System.out.println(order.value);

15: } }

A. tacb

B. tacf

C. tacbf

D. tacfb

E. tacftacb

F. The code does not compile.

G. An exception is thrown.

24. Which of the following will compile when inserted in the following code? (Choose

all that apply)

public class Order3 {

final String value1 = "1";

static String value2 = "2";

String value3 = "3";

{

// CODE SNIPPET 1

}

static {

// CODE SNIPPET 2

}

}

A. value1 = "d"; instead of // CODE SNIPPET 1

B. value2 = "e"; instead of // CODE SNIPPET 1

C. value3 = "f"; instead of // CODE SNIPPET 1

D. value1 = "g"; instead of // CODE SNIPPET 2

E. value2 = "h"; instead of // CODE SNIPPET 2

F. value3 = "i"; instead of // CODE SNIPPET 2

Review Questions 229

c04.indd 1½ 4/2014 Page 229

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

public class Create {

Create() {

System.out.print("1 ");

}

Create(int num) {

System.out.print("2 ");

}

Create(Integer num) {

System.out.print("3 ");

}

Create(Object num) {

System.out.print("4 ");

}

Create(int... nums) {

System.out.print("5 ");

}

public static void main(String[] args) {

new Create(100);

new Create(1000L);

}

}

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 constructor Create(int num).

F. The code prints 4 4 if you remove the constructor Create(int num).

G. The code prints 5 4 if you remove the constructor Create(int num).

26. What is the result of the following class?

1: import java.util.function.\*;

2:

3: public class Panda {

4: int age;

5: public static void main(String[] args) {

6: Panda p1 = new Panda();

7: p1.age = 1;

8: check(p1, p -> p.age < 5);

230 Chapter 4 ■ Methods and Encapsulation

c04.indd 1½ 4/2014 Page 230

9: }

10: private static void check(Panda panda, Predicate<Panda> pred) {

11: String result = pred.test(panda) ? "match" : "not match";

12: System.out.print(result);

13: } }

A. match

B. not match

C. Compiler error on line 8.

D. Compiler error on line 10.

E. Compiler error on line 11.

F. A runtime exception is thrown.

27. What is the result of the following code?

1: interface Climb {

2: boolean isTooHigh(int height, int limit);

3: }

4:

5: public class Climber {

6: public static void main(String[] args) {

7: check((h, l) -> h.append(l).isEmpty(), 5);

8: }

9: private static void check(Climb climb, int height) {

10: if (climb.isTooHigh(height, 10))

11: System.out.println("too high");

12: else

13: System.out.println("ok");

14: }

15: }

A. ok

B. too high

C. Compiler error on line 7.

D. Compiler error on line 10.

E. Compiler error on a different line.

F. A runtime exception is thrown.

28. Which of the following lambda expressions can fill in the blank? (Choose all that apply)

List<String> list = new ArrayList<>();

list.removeIf(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

Review Questions 231

c04.indd 1½ 4/2014 Page 231

A. s -> s.isEmpty()

B. s -> {s.isEmpty()}

C. s -> {s.isEmpty();}

D. s -> {return s.isEmpty();}

E. String s -> s.isEmpty()

F. (String s) -> s.isEmpty()

29. Which lambda can replace the MySecret class to return the same value? (Choose

all that apply)

interface Secret {

String magic(double d);

}

class MySecret implements Secret {

public String magic(double d) {

return "Poof";

}

}

A. caller((e) -> "Poof");

B. caller((e) -> {"Poof"});

C. caller((e) -> { String e = ""; "Poof" });

D. caller((e) -> { String e = ""; return "Poof"; });

E. caller((e) -> { String e = ""; return "Poof" });

F. caller((e) -> { String f = ""; return "Poof"; });

Chapter 4: Methods and Encapsulation

1. B, C. 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 correct, creating a method with default access and the optional specifier

final. Since default access does not require a modifier, we get to jump right to final.

Chapter 4: Methods and Encapsulation 343

bapp01.indd 1½ 4/2014 Page 343

Option A is incorrect because default 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.

2. 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.

3. A, C, D. Options A and C are correct because a void method is allowed to have a

return statement as long as it doesn’t try to return a value. Options B and G do not

compile because null requires a reference object as the return type. void is not a reference object since it is a marker for no return type. int is not a reference object since it

is a primitive. 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.

4. A, B, G. Options A and B are correct because the single vararg parameter is the last

parameter declared. Option G is correct because it doesn’t use any vararg parameters

at all. Options C and F are incorrect because the vararg parameter is not last. Option

D is incorrect because two vararg parameters are not allowed in the same method.

Option E is incorrect because the ... for a vararg must be after the type, not before it.

5. D, G. Option D passes the initial parameter plus two more to turn into a vararg array

of size 2. Option G passes the initial parameter plus an array of size 2. Option A does

not compile because it does not pass the initial parameter. Options E and F do not

compile because they do not declare an array properly. It should be new boolean[]

{true}. Option B creates a vararg array of size 0 and option C creates a vararg array of

size 1.

6. D. Option D is correct. This is the common implementation for encapsulation by setting all fields to be private and all methods to be public. Option A is incorrect because

protected access allows everything that package private 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 private access applies to the whole package. Option E is incorrect because Java

has no such capability.

7. B, C, D, F. The two classes are in different packages, which means private access and

default (package private) access will not compile. Additionally, protected access will

not compile since School does not inherit from Classroom. Therefore, only line 8 will

compile because it uses public access.

8. B, C, E. Encapsulation requires using methods to get and set instance variables so

other classes are not directly using them. Instance variables must be private for this

to work. Immutability takes this a step further, allowing only getters, so the instance

variables do not change state.

344 Appendix A ■ Answers to Review Questions

bapp01.indd 1½ 4/2014 Page 344

9. C, E. Option A is incorrect because the property is of type boolean and getters must

begin with is for booleans. Options B and D are incorrect because they don’t follow

the naming convention of beginning with get/is/set. Options C and E follow normal

getter and setter conventions.

10. B. Rope runs line 3, setting LENGTH to 5, then immediately after runs the static initializer, which sets it to 10. Line 5 calls the static method normally and prints swing.

Line 6 also calls the static method. Java allows calling a static method through an

instance variable. Line 7 uses the static import on line 2 to reference LENGTH.

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. If line

10 is removed, the code works. It does not throw a NullPointerException on line 16

because play() is a static method. Java looks at the type of the reference for rope2 and

translates the call to Rope.play().

12. 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 one object update this common static variable.

13. E. static final variables 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.

14. 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 only do a static import 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.

15. 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

intObjectObject and the correct answer is option E.

16. 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. y 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.

17. B, D, E. Since Java is pass-by-reference, assigning a new object to a does not change the

caller. Calling append() does affect the caller because both the method parameter and

Chapter 4: Methods and Encapsulation 345

bapp01.indd 1½ 4/2014 Page 345

caller have a reference to the same object. Finally, returning a value does pass the reference to the caller for assignment to s3.

18. C, G. Since the main() method is in the same class, it can call private methods in the

class. this() may only be called as the first line of a constructor. this.variableName

can be called from any instance method to refer to an instance variable. It cannot be

called from a static method because there is no instance of the class to refer to. Option

F is tricky. The default constructor is only written by the compiler if no user-defined

constructors were provided. this() can only be called from a constructor in the same

class. Since there can be no user-defined constructors in the class if a default constructor was created, it is impossible for option F to be true.

19. A, G. Options B and C don’t compile because the constructor name must match the

classname. Since Java is case sensitive, these don’t match. Options D, E, and F all compile and provide one user-defined constructor. Since a constructor is coded, a default

constructor isn’t supplied. Option G defines a method, but not a constructor. Option A

does not define a constructor, either. Since no constructor is coded, a default constructor is provided for options A and G.

20. E. Options A and B will not compile because constructors cannot be called without

new. Options C and D will compile but will create a new object rather than setting the

fields in this one. Option F will not compile because this() must be the first line of a

constructor. Option E is correct.

21. C. Within the constructor numSpots refers to the constructor parameter. The instance

variable is hidden because they have the same name. this.numSpots tells Java to use

the instance variable. In the main() method, numSpots refers to the instance variable.

Option A sets the constructor parameter to itself, leaving the instance variable as 0.

Option B sets the constructor parameter to the value of the instance variable, making

them both 0. Option C is correct, setting the instance variable to the value of the constructor parameter. Options D and E do not compile.

22. E. On line 3 of OrderDriver, we refer to Order for the first time. At this point the statics in Order get initialized. In this case, the statics are the static declaration of result

and the static initializer. result is u at this point. On line 4, result is the same

because the static initialization is only run once. On line 5, we create a new Order,

which triggers the instance initializers in the order they appear in the file. Now result

is ucr. Line 6 creates another Order, triggering another set of initializers. Now result

is ucrcr. Notice how the static is on a different line than the initialization code in

lines 4–5 of Order. The exam may try to trick you by formatting the code like this to

confuse you.

23. A. Line 4 instantiates an Order. Java runs the declarations and instance initializers first

in the order they appear. This sets value to tacf. Line 5 creates another Order and

initializes value to tacb. The object on line 5 is stored in the same variable line 4 used.

This makes the object created on line 4 unreachable. When value is printed, it is the

instance variable in the object created on line 5.

346 Appendix A ■ Answers to Review Questions

bapp01.indd 1½ 4/2014 Page 346

24. B, C, E. value1 is a final instance variable. It can only be set 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. value2 is a static variable. Both

instance and static initializers are able to access static variables, making options B

and E correct. value3 is an instance variable. Options D and F do not compile because

a static initializer does not have access to instance variables.

25. A, E. The 100 parameter is an int and so calls the matching int constructor. When

this constructor is removed, Java looks for the next most specific constructor. Java prefers autoboxing to varargs, and so 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 constructor for Object is called.

26. A. This code is correct. Line 8 creates a lambda expression that checks if the age is less

than 5. Since there is only one parameter and it does not specify a type, the parentheses

around the type parameter are optional. Line 10 uses the Predicate interface, which

declares a test() method.

27. C. The interface takes two int parameters. The code on line 7 attempts to use them as

if one is a StringBuilder. It is tricky to use types in a lambda when they are implicitly

specified. Remember to check the interface for the real type.

28. A, D, F. removeIf() expects a Predicate, which takes a parameter list of one parameter using the specified type. Options B and C are incorrect because they do not use the

return keyword. It is required inside braces for lambda bodies. Option E is incorrect

because it is missing the parentheses around the parameter list. This is only optional

for a single parameter with an inferred type.

29. A, F. Option B is incorrect because it does not use the return keyword. Options C, D,

and E are incorrect because the variable e is already in use from the lambda and cannot be redefined. Additionally, option C is missing the return keyword and option E is

missing the semicolon.