# Objects and Inheritance – Answers

Question 1

b and f

Two objects and three reference variables are created by the code. Objects are typically created by using the new operator. Declaration of a reference variable creates a variable regardless of whether a reference value is assigned to it or not.

Question 2

a and b

The extends clause is used to specify that a class extends another class. A subclass can be declared abstract regardless of whether the superclass was declared abstract. Private, overridden, and hidden members from the superclass are not inherited by the subclass. A class cannot be declared both abstract and final, since an abstract class needs to be extended to be useful and a final class cannot be extended. The accessibility of the class is not limited by the accessibility of its members. A class with all the members declared private can still be declared public.

Question 3

**d and f**

Given the declaration class B extends A {...} we can conclude that class B extends class A, class A is the superclass of class B, class B is a subclass of class A, and class B inherits from class A, which means that objects of class B will inherit the field value1 from class A.

Question 4

c

The local variable of type float will remain uninitialized. Fields receive a default value unless explicitly initialized. Local variables remain uninitialized unless explicitly initialized. The type of the variable does not affect whether a variable is initialized or not.

Question 5

**c**

Strings are objects. The variables a, b, and c are references that can denote such objects. Assigning to a reference only changes the reference value. It does not create a copy of the source object or change the object denoted by the old reference value in the destination reference. In other words, assignment to references only affects which object the destination reference denotes. The reference value of the "cat" object is first assigned to variable a, then to variable b, and later to variable c. The program prints the string denoted by the variable c, which is "cat".

Question 6

c

The declaration abstract int t; is not legal. Keywords static and final are valid modifiers for both field and method declarations. The modifiers abstract and native are only valid for methods.

Question 7

a and e

The var-args syntax f(type… params) allows a variable number of parameters of the given type. The number is from zero to many. (a) allows zero or more ints. (e) requires an int argument to match the first parameter, followed by zero or more arguments to match the 2nd parameter. These will therefore accept method calls doIt(1) and doIt(1,2).

(b) requires an argument of type int array, so neither method invocation will compile. (c) is illegal; the ellipsis (3 dots) must follow the type, not the parameter name. (d) is illegal; the var-args parameter must appear last in the method signature. (f) is illegal; extra dimensions (the array brackets) are not allowed on var-arg parameters.

Question 8

a, c, and d

Fields in interfaces declare named constants, and are always public, static, and final. None of these modifiers are mandatory in a constant declaration. All named constants must be explicitly initialized in the declaration. Private members, whether fields or methods, are never allowed in an interface. All members of an interface must be public.

Question 9

b

In a subclass without any declared constructors, the implicit default constructor will call super(). The use of the super() and this() statements are not mandatory as long as the superclass has a default constructor. If neither super() nor this() is declared as the first statement in the body of a constructor, then super() will implicitly be the first statement. A constructor body cannot have both a super() and a this() statement. Calling super() will not always work, since a superclass might have neither a default nor a no-arg constructor.

Question 10

e

The program will print 2 when System.out.println(ref2.f()) is executed. The object referenced by ref2 is of class C, but the reference is of type B. Since B contains a method f(), the method call will be allowed at compile time. During execution it is determined that the object is of class C, and dynamic method lookup will cause the overridden method in C to be executed. Dynamic resolution of overridden methods is always.

Question 11

a and b

When overriding, the return type must be the same type or a subclass of the overridden method’s return type. The return type of X.getIt() is B. Both (a) and (b) fulfill the rule. (c) and (d) cause a compile error wrong because they return an A – which is the superclass of B. (e) returns the valid type C in its signature, but tries to return a B, causing a compile error since B is the superclass of C; the IS-A test fails!

Question 12

c

The line void k() { i++; } can be re-inserted without introducing errors. Re-inserting line (1) will cause the compilation to fail, since MyOtherClass will try to override a final method. Re-inserting line (2) will fail, since MyOtherClass will no longer have a default constructor. The main() method needs to call the default constructor. Re-inserting line (3) will work without any problems, but re-inserting line (4) will fail, since the method will try to access a private member of the superclass.