# Declarations and Access Control – Answers

## Section 1: Classes

Question 1

c d e

If a class C is declared as a member of an enclosing class then C may be declared using no access modifier or any of the three access modifiers, *private*, *protected* or *public*. However, if class C is not a local class, anonymous class or a member of an enclosing class or interface; then C may be declared with the *public* modifier or with package access (i.e. no modifier). The other two access modifiers, *private* and *protected*, are not applicable to any class that is not a member class. The class declaration, Class Basics4 {}, generates a compile-time error, because all of the letters of the reserved word *class* must be lower case.

Question 2

a

Only one class in a source code file can be declared *public*. The other classes may not be *public*. Therefore, the declarations for classes Basics2, Basics3 and Basics4 generate compile-time errors.

Question 3

a d f

public abstract final

The access modifiers, *protected* and *private*, can be applied to a class that is a member of an enclosing class, but cannot be applied to a local class or a class that is not nested inside another class. The *static* modifier can be applied to a class that is a member of an enclosing class, but cannot be applied to a local class or a class that is not nested inside another class. The *public* modifier can be applied to a top level class to allow the class to be accessed from outside of the package. The *abstract* modifier prevents the class from being instantiated. An abstract class may include zero, one or more abstract methods. The *final* modifier prevents a class from being extended.

Question 4

a

abstract class A {} // 1

The modifiers, *private* and *static*, can be applied to a nested class, but cannot be applied to a class that is not nested. A class that is not nested can have *public* or package access, but not *private*. The *transient* modifier cannot be applied to any class; because it is a field modifier.

Question 5

c. The variable should be marked protected

Question 6

e None of the above

All fields within an interface are implicitly *public*, *static* and *final*. Use of these modifiers is redundant but legal. No other modifiers can be applied to a field declaration within an interface.

Question 7

b

class A implements Z {void m1() {}} // 2

All methods declared within an interface are implicitly *abstract* and *public*. Although the *abstract* and *public* modifiers can legally be applied to a method declaration in an interface, the usage is redundant and is discouraged. Methods declared within an interface are implicitly *public* even if the modifier *public* is omitted from the declaration. Within the body of a class declaration, an attempt to implement the method using a weaker access privilege (*private*, *protected* or package access) results in a compile-time error. An abstract class that implements an interface is free to override any of the inherited method declarations with another abstract method declaration.

Question 8

c Compile-time error

Fields declared within an interface are implicitly *public*, *final*, and *static*. A compile-time error is generated in response to the attempt to increment the value of i.

Question 9

b) False

## Section 2: Arrays

Question 1

a e

An array creation expression must have either a dimension expression or an initializer. If both are present, then a compile-time error is generated. Similarly, if neither is present, then a compile-time error is generated. If only the dimension expression is present, then an array with the specified dimension is created with all elements set to the default values. If only the initializer is present, then an array will be created that has the required dimensions to accommodate the values specified in the initializer. Java avoids the possibility of an incompatible dimension expression and initializer by not allowing both to appear in the same array creation expression. A compile-time error is generated by the array creation expression for a1, because it needs either a dimension expression or an initializer. A compile-time error is generated for a5, because either the dimension expression or the initializer must be removed.

Question 2

b (Prints: 0,0,0.0,null)

Each array contains the default value for its type. The default value of a primitive *byte* or a primitive *long* is printed as 0. The default value of a primitive *float* is printed as 0.0. The default value of an *Object* is null and is printed as null.

## Section 3: Keywords

Question 1

d (import, break, double, exception, throws )

The word *exception* is not a Java keyword. The words *import*, *break*, *double* and *throws* are Java keywords.

Question 2

b e g h k (continue new finally const do)

Question 3

d f h (4, 6, 8)

The first letter of an identifier can be any Unicode JLS 3.1 character that is a Java letter. The first letter can not be a number. For historical reasons, the dollar sign $ and underscore \_ are considered Java letters along with many currency symbols in use in the world today.

## 