Unit 05 Review Questions 1

©2024 Chris Nielsen – *www.nielsenedu.com*

1. Answer each in complete sentences with proper grammar. Points will be given for neatness.

   a) In the context of object-oriented programming, what is a ***class***?

   A ***class*** is a template (or *blueprint*) for creating objects. A ***class*** contains ***fields***, which define the *state*, and ***methods***, which define the *behavior* of the object.

   b) In the context of object-oriented programming, what is a ***field***?

   A ***field*** (or ***attribute***) is a variable defined at the top level within a class. The *state* of an object is defined by the values of its fields.

   c) In the context of object-oriented programming, what is a ***method***?

   A ***method*** (or ***operation***) is code that operates on the object it is associated with. Methods describe the *behavior* of an object.

   d) Explain how the **`private`** modifier affects access to *fields*?

   The `private` modifier makes a field accessible only from within the same class. Other classes cannot directly read or modify a private field, but may still call public methods that are within the same class as the private field in order to modify it.

   e) Explain how the **`private`** modifier affects access to *methods*?

   The `private` modifier prevents other classes from calling a method. These methods can only be called from within the class where they are defined.

   f) What is the purpose of the ***constructor*** in Java?

   A ***constructor*** is a special method that initializes the fields of an object when the object is created.

   g) What is the purpose of a **`toString`** method in Java?

   The `toString` method provides a `String` representation of the object, summarizing key fields of the object. Typically this is for printing or logging.

   h) What is the purpose of an ***accessor*** method in Java?

   An ***accessor***, or getter, method allows controlled read access to private fields.

   i) What is the purpose of an ***mutator*** method in Java?

   An ***mutator***, or getter, method allows controlled modification access to private fields.

j) What is the benefit of a ***mutator*** method over making a field `public`?

Using an accessor method prevents direct field modification, which ensures data integrity by allowing validation checks when the field is set.

k) When reading the declaration of a field in Java, how are *class* fields distinguished from *instance* fields?

A class field has the modifier `static`, while an instance field omits this modifier.

2. Consider the following class definition:

```
1 public class Vector {
2    public double x;
3    public double y;
4 }
```

a) How does Java treat classes that have no explicit constructor defined?

When a class does not have a constructor, Java will automatically generate a default constructor that takes no arguments. This constructor will initialize all fields to their default values (zero for numbers, `false` for `boolean`, etc.).

b) Given the above `Vector` class, write a line of code that will create a new object, `v`, of type `Vector`.

```
Vector v = new Vector();
```

c) If a constructor for the `Vector` class were to be explicitly defined, and take two parameters, each parameter of type `double`, in order to initialize the fields `x` and `y`, the line of code written in part (b) would no longer work. Why?

Java will not automatically generate the default constructor if any constructor with any number of parameters is explicitly defined. Therefore, if a two-parameter constructor is defined, then it will be an error to try to call a zero-parameter constructor if it is not explicitly defined.

d) If a two-parameter constructor were explicitly defined, as in part (c), but we still wish to be able to call a zero-parameter constructor, as in part (b), what is the solution?

If we wish to have a two-parameter constructor and still use the zero-parameter constructor, we must explicitly define the zero-parameter constructor.