

Sample Quiz Solutions

1. **True** (a) or **False** (b): In Java, there can be, at most, only one `main()` method in each project
2. The default data type of a numeric value (to the right of the '=' sign) in a definition like:
`short mol = 42;` is:
(a) byte (b) short (c) integer (d) long (e) none of the above
3. .jar files shown in the Package Explorer contain _____, which in turn contain _____
(a) packages, bytecode (b) libraries, packages (c) packages, classes
(d) src, sourcecode (e) none of the above
4. **True** (a) or **False** (b): `static` methods can only change `static` variables of (or within) the same class, along with instantiated variables within a class
5. Which one of the following keywords does not limit the accessibility of an instantiated class's methods to access from other objects?
(a) private (b) protected (c) package (d) public (e) none of the above

6. Which symbol is used to indicate that a method is *private* in UML?
(a) - (b) + (c) # (d) @ (e) none of the above
7. Which one of the following keywords is used to signal that one class inherits its non-private members from another?
(a) extends (b) inherits (c) super (d) subclass (e) none of the above
8. If A is a class derived from B via an *is a* relationship (A *is a* B, using the keyword from question 7 above) which one of the following terms does *not* describe the relationship of A to B?
(a) subclass (b) derived class (c) base class (d) child class
(e) none of the above
9. The first thing loaded when a class is *instantiated* is:
(a) The Object class's members (b) The default no-arg constructor
(c) the class's static members (d) the constructor for the object itself
(e) none of the above
10. **True** (a) or **False** (b): The second (or middle) compartment of a UML class diagram may be omitted if the fields of that class are private

11. Which one of the following will **NOT** correctly declare an array of 3 integers?
(i.e. it will flag a compile-time error):

- (a) `int[] myInts = new int[3]{1, 2, 3};`
- (b) `int[] myInts = new int[] {1, 2, 3};`
- (c) `int[] myInts = new int[3];`
- (d) `int[] myInts = {1, 2, 3};`
- (e) none of the above

Part B: Terminology – worth 1 mark each

FILL IN THE SINGLE BEST ANSWER—ONE WORD OR SYMBOL IN THE EACH SPACE PROVIDED BELOW.

USE ONE WORD ONLY IN EACH SPACE; DO NOT USE ACRONYMS

12. Fields and methods are collectively referred to as the members of an object.
13. One of the fundamental features of OOP is encapsulation, in which fields and methods are bound together.
14. If a .java program is located in the folder
C:\Users\username\workspace\Asmt1\package2\src, then its compiled bytecode should appear in C:\Users\username\workspace\Asmt1\ package2 \ bin.
15. Static members are often times referred to as class members.
16. Objects cannot be instantiated from abstract classes; they can only be instantiated from concrete classes
17. In an array definition like: `datatype[] identifier = new datatype[arraysize];`
`arraysize` indicates the total number of elements in the array

In file DemoSpeed.java: 14

```
import Scanner; 1
public Class DemoSpeedPerHour{
    public void main(String[] args){ 2
        Scanner in = new Scanner(system.in); 3
        System.out.println("Enter distance"); 4
        long distance = in.nextLong; 5
        GetSpeed gs = new GetSpeed(distance); 6
        int speed = gs.calcVelocity(); 7
        System.out.printf("%s", speed); 8
    }
}
```

In file GetSpeed.java:

```
private class GetSpeed{
    private long distance;
    private int hours;

    void GetSpeed(int d){distance = d; 9
    void GetSpeed(int d, int seconds){
        distance = d;
        hours = seconds/3600;
    } 10
    public float calcVelocity(){ 11
        return( (float)(distance/hours)); 12
    } 13
}
```

In file DemoSpeed.java:

```
import Scanner;
public Class DemoSpeedPerHour{
    public void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter distance");
        long distance = in.nextLong();
        GetSpeed gs = new GetSpeed(distance);
        int speed = gs.calcVelocity();
        System.out.printf("%s", speed);
    }
}
```

In file GetSpeed.java:

```
private class GetSpeed{
    private long distance;
    private int hours;

    void GetSpeed(int d){distance = d;
    void GetSpeed(int d, int seconds){
        distance = d;
        hours = seconds/3600;
    }
    public float calcVelocity(){
        return( (float)(distance/hours));
    }
}
```

1. Should be `java.util.Scanner`
2. Class is should be `class`
3. Missing `static` from method header
4. Should be `System.in`
5. `nextLong` is a method, so should be `nextLong()`
6. Attempt to pass long value into int parameter; downcast required
7. `calcVelocity()` returns a float, not an int. Hence speed should be declared as a float, not an int.
8. Type mismatch; speed is declared an int, but `printf()` uses `%s`, indicating String output (should be `%d`)
9. Missing `;` at end of statement
10. Constructors return nothing, not even void
11. hours needs to be a floating point value, otherwise there will be a significant round-off error in this calculation (3599 seconds = 0 hours)
12. Problem if first constructor is used, since hours will be 0 by default.
13. distance needs to be cast as float first (inside brackets) otherwise significant loss of accuracy
14. File name must be same as class name

In file DemoSpeed.java:

```
import Scanner;
public Class DemoSpeedPerHour{

    public void main(String[] args){
        Scanner in = new Scanner(system.in);
        System.out.println("Enter distance");
        long distance = in.nextLong;
        GetSpeed gs = new GetSpeed(distance);
        int speed = gs.calcVelocity();
        System.out.printf("%s", speed);
    }
}
```

In file GetSpeed.java:

```
private class GetSpeed{
    private long distance;
    private int hours;

    void GetSpeed(int d){distance = d}
    void GetSpeed(int d, int seconds){
        distance = d;
        hours = seconds/3600;
    }
    public float calcVelocity(){
        return( (float)(distance/hours));
    }
}
```

Note that the following are NOT errors, since they would neither trigger compile time errors nor would they lead to run-time errors:

- Not having a no-arg GetSpeed constructor (*this is only a problem if you extend GetSpeed in a subclass*)
- Using the hours to calculate the speed, rather than seconds or minutes (*it's the designer's choice, not yours*)
- GetSpeed doesn't actually calculate the speed, it merely sets properties for the class (*which is what constructors are supposed to do*)
- Not having an access modifier (like public) on the constructors (*the default applies: package private, i.e. public inside the package*)
- Not checking for divide-by-zero errors (*again, this is the designer's choice, not yours. Only the first GetSpeed() constructor is capable of causing problems. So your comments should address the actual source of the error at this point, not the fact there's no check for it in the code later on.*)

Remember: do not list as errors things which you disagree with (including poor design). Only list those items which are truly in error.

19. In the three boxes below, show the UML class diagram for the `GetSpeed` class from the previous question, being clear to indicate all properties and methods (including private ones), using the correct symbols and the format for UML

GetSpeed
-distance: long -hours: int
+GetSpeed(d: int) +GetSpeed(d: int, seconds: int) +calcVelocity(): float

Note: this assumes corrected constructor definition