Quiz Solutions

Note: these answers are based on the 'C' version of the quiz. If you wrote the 'D' version, the same questions appear, but in a different order. In some cases, the wording between exams may be different, so check the wording carefully before reporting errors.

Part A: Multiple Choice Questions – worth 1 mark each
Choose the single best answer for each question.
ANSWER ON THE SCANTRON

2. **TRUE** (a) or **FALSE** (b): A class's name will *always* be the same as the java file in which the class is loaded (minus the *.java* extension, of course).

3. **TRUE** (a) or **FALSE** (b): A constructor cannot be private

4. When a constructor in a subclass is tied to its superclass constructor (using `super()`), it is said to be:

   (a) overloaded (b) overridden (c) chained (d) static (e) none of the above

5. Which one of the following statements is **TRUE** of *encapsulation*:

   (a) It ensures that certain members of an object will be hidden from the user's view, accessible only through public getters and setters.
   (b) It ensures that each object can only be instantiated from a class with a single parent class
   (c) It means that the state and behavior of an object are declared together in the class.
   (d) It allows the non-private features of the superclass to be used in any of its subclasses.
   (e) none of the above

6. When the following code fragment is executed

```
long num = 720;
float ratio = num/10.0;
```

the value stored in `ratio` will be

(a) 70.0 (b) 70 (c) 72 (d) 72.0 (e) not calculated, since an error occurs first

7. For the class `Counter` having the method

```
public static void setCounter(int newCtr) {…}
```

whenever an integer value is passed as a parameter to `setCounter` (for example, the value 5 in `Counter.setCounter(5)`), the method's parameter, `int newCtr`, acts as a:

(a) assignment (b) static variable (c) definition (d) declaration (e) none of the above

8. In question 7 above, which one of the following terms would ***NOT*** be used to describe `setCounter()`

(a) static member (b) instance method (c) class method (d) mutator (e) none of the above

9. In debug mode, to *step over a line*, you would use:

a)  (or Ctl-F11) b)  (or F5) c)  (or F6) d)  (or F8) e) none of the above

10. **TRUE** (a) or **FALSE** (b): The `File` object cannot be used to add new contents to a file.

11. If the access modifier is omitted from a method's declaration, then its visibility is limited to

(a) the class (b) the package (c) the project (d) subclasses
(e) it can't be done; methods must always have an access modifier

12. The third (last) compartment of the UML class diagram is used to indicate that class's
(a) name (b) fields (c) methods (d) access modifiers (e) none of the above

13. If a class contains the following import statements

```
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.ScrollPane.ScrollBarPolicy;
```

then which *one* of the following is **TRUE** of the statement

```
import javafx.scene.control.*;
```

(a) this statement can be used to replace all of the above `import` statements
(b) this statement can ***NOT*** be used to replace any of the above `import` statements
(c) this statement can be used to replace all of the above `import` statements except the last one
(d) this statement can be used to replace the first two `import` statements only
(e) this is not a viable `import` statement; it will trigger a compile-time error in Eclipse

14. **TRUE** (a) or **FALSE** (b): `Scanner` can be used to read the contents of a text file into a program

15. Which one of the following code fragments does ***NOT*** result in an infinite loop (leading to a possible run-time error)?

(a) `for(;;){}`
(b) `while(true){}`
(c) `do { } while (false);`
(d) `for(int x = 0; x < 1; --x){}`
(e) none of the above

16. **TRUE** (a) or **FALSE** (b): A class cannot instantiate a new instance of itself if it contains static members.

17. Which one of the following symbols is used to indicate an inheritance relationship between a subclass and its superclass (i.e. an '*is a*' relationship):

(a)  (b)  (c)  (d)  (e) none of the above

18. **TRUE** (a) or **FALSE** (b): A class cannot contain more than one `main()` method.

19. Which one of the following is not part of the signature of a method?
(a) the name of the method (b) the names of the variables used in the parameter list
(c) the number of parameters (d) the type of each parameter (e) the order of the parameters

20. Which symbol is used to indicate that a method is `protected` in UML?

(a) **-** (b) **+** (c) **#** (d) @ (e) none of the above

21. In a UML diagram, constants are identified by being:

a) underlined b) *italicized* c) **bold-face** d) CAPITALIZED e) none of the above

22. Which one of the following terms is not used to refer to a relationship between objects in OOP?

a) association b) aggregation c) composition d) inheritance e) none of the above

23. The `package` and `import` statements are responsible for creating a separate

namespace for each class or object identifier used in a program.

24. To run a java class on any device, one must first install the *Java*

Virtual *Machine* (*3 words, no acronyms*) , which is software that acts like hardware specifically designed to execute Java bytecode via its JRE (Java Runtime Environment).

25. Java only allows *single* inheritance ; This prevents what is known as the diamond problem

26. The UML diagram showing a family of interconnected sub- and super-classes (such the one on page 5) is an example of a(n) *class (or object)* *hierarchy* (2 words)

Part C: Written Answers – worth 9 marks total

```
public class CalorieCounterDemo {

   public static void main(String args) {
      Apples[] apples = new Apple[3];
      apples[1] = new Apple(0.4f);
      apples[2] = new Apple();
      apples[3] = new Apple(0.3f);

      float totCalories = 0;
      for (int ctr=0; ctr < apples.length; ctr++)
           totCalories += apples[ctr].getCalories();
      System.out.println("Total calories in these "+
        apples.length + " apples is " +
        totCalories + " Calories");
   }
}
```

1
2
3

1. Should be String[]
2. Should be Apple, not Apples, to be consistent with the data type used throughout the class
3. Apple array index is off by 1; should start at 0 and end at 2.

```
public class Apple extends Fruit {

   private float weight = 100.0;

   public float getWeight() {return weight;}
   public void setWeight(float weight) {
     if (getWeight() <= 0)
      System.out.println("Weight cannot be" +
      " less than or equal to zero; resetting" +
      " to default value 100 grams");
         this.weight = (weight<=0)?weight:100;
   }

   public Apple() {super();}
   public Apple(float pS) {super(pS);}

   public float getCalories() {
     return (getWeight() *
     3*(getSugarContent())+getProteinContent());
   }
}
```

4 5 6 7

4. Number will default to double, which cannot fit into float
5. getWeight() checks the currently-stored value of weight, when it needs to check the weight just entered.
6. Assignment of weight is backwards; should be if weight less than or equal to zero, then reset to 100
7. Only multiplying sugarContent by 3; inside parenthesis should be around the whole equation

```
public class Fruit extends Food {
   private boolean sweet = true;
   private float proteinContent, sugarContent;

   public boolean isSweet() {return sweet;}
   public void setSweet(boolean s) {sweet = s;}

   public float getProteinContent() {
      return proteinContent;}
   public void setProteinContent(float protein){
      proteinContent = protein;}

   public float getSugarContent() {
      return sugarContent;}
   public void setSugarContent(float s) {
      sugarContent = s;}

   public Fruit () {this(0.1f);}
   public Fruit (float pC) {this(pC, 0.3);}
   public Fruit(float pC, float pS) {
     setProteinContent(pC);
     float tempFibre = getFibreContent() -
     (getProteinContent() + getSugarContent());
     setSugarContent(isSweet?pS+0.30:pS);
         //incr sugar by 30% if sweet
     setFibre(tempFibre);
   }
}
```

8
9
11
10
12

8. One-arg Fruit constructor is passed the sugar content (using super(pS) from Apple, but uses this as the protein content (pC) in Fruit
9. getFibreContent() does not exist; it should be getFibre(), as declared in Food class
10. isSweet is a method, therefore needs ()
11. 0.30 is a double by default, hence ps+0.3 returns a double, while setSugarContent() expects a float
12. Setting sugar content after the calculation, so the fibre content calculated will not be accurate

```
public class Food extends Fruit {
  private float fibreContent = 1;
  /// % fibre as decimal value; 1 = 100%
   public float getFibre() {return fibreContent};
   public void setFibre(float fib) {
      fibreContent = fibre;
   }
}
```

13. Food is the superclass for Fruit, not the other way around
14. Semicolon should be inside brace, not after
15. Should use fib or fibre consistently, otherwise value not set correctly.