Quiz Answers

Note: these answers are based on the 'A' version of the quiz. If you wrote the 'B' 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.

Also note: when this .pptx file is run as a presentation, the answers appear after you select the spacebar. It can thus be used to practice for future exams.

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): You can only chain between the constructors of a class, not its methods

3. Which one of the following is *NOT* part of the signature of a method?

   (a) the name of the method (b) the number of parameters (c) the type of each parameter

   (d) the identifier of the parameters (e) the order of the parameters

4. **TRUE** (a) or **FALSE** (b): instance methods are loaded before static methods.

5. Which one of the following OOP terms best describes the relationship between the College and its cafeteria?
(a) association (b) composition (c) inheritance (d) regurgitation (e) none of the above

6. **TRUE** (a) or **FALSE** (b): Whenever a class is contained in a named package (i.e. the class is *NOT* in the default package, which has no name), the package name must be included as the first line of that class.

7. Which one of the following is *NOT* true of the use of the keyword `final`?
(a) When used with a class, it prevents that class from being subclassed through inheritance
(b) When used in the declaration of a method, it ensures that method cannot be overridden
(c) When used in the declaration of a field of a primitive data type, it ensures the value of the field can be set only once, making it essentially constant
(d) When used with a variable that holds an instantiated constructor, it ensures the variable cannot be used to load a new instance of the same object
(e) none of the above

8. Which one of the following statements is true of the `@Override` annotation?
(a) it is required whenever a subclass method needs to override a superclass method
(b) `Override` is a keyword in the java language
(c) if it precedes a subclass method with the same signature as an existing superclass method, it is not essential
(d) it cannot be used if the method that follows the annotation is declared `final`
(e) none of the above

9. Which one of the following symbols would be used to indicate that one class has a casual association with another class, that is, that is causes the second class to happen in a one-off relationship?
(a)  (b)  (c)  (d)  (e) none of the above

10. Which one of the following `Object` class methods is called just prior to garbage collection?

(a) `toString()` (b) `finalize()` (c) `clone()` (d) `equals()` (e) none of the above

11. In debug mode, to *step return* from a method, you would click on:

(a) (or F8) (b) (or F5) (c) (or F6) (d) (or F7) (e) none of the above

12. **TRUE** (a) or **FALSE** (b): Whenever a subclass is instantiated, an instance of the `Object` class will be loaded first, since the `Object` class is the ultimate parent of any class.

CST8284 – Object-Oriented Programming (Java) – Sec. 310

Hybrid Quiz

13. **TRUE** (a) or **FALSE** (b): *Implicit* type conversion occurs whenever you attempt to assign a smaller type value to a variable declared as a larger type.

14. Which one of the following is *NOT* a requirement for a valid identifier in Java?

(a) it cannot begin with a number
(b) it cannot begin with an underscore, _
(c) it cannot be one of the reserved/key words in java, such as `if`, `new`, `break`, `null`, etc.
(d) an identifier written entirely in upper case characters is different from the same identifier written in lower case characters.
(e) none of the above

15. **TRUE** (a) or **FALSE** (b): the middle compartment of the UML class diagram may be omitted.

16. Which *one* of the following is used in UML diagrams to signal that a field is static?

(a) CAPITALIZE (b) underline (c) *italics* (d) **boldface** (e) none of the above

17. What will be the result of the following calculation in Java?

```
int x = 34;
float y = (x/10d) + 3.6f;
```

(a) `y=6.6` (b) `y=7.0` (c) type mismatch error (d) run time error (e) none of the above

18. **TRUE** (a) or **FALSE** (b): In file IO, the term 'output' is used to indicate that information is taken from one or more variables in a program and loaded into a file.

19. Which one of the following operations is a `File` object *NOT* capable of performing (that is, *without* the help of additional operations from other classes)?

(a) an operation that returns the number of Strings in a text file
(b) an operation that returns the number of bytes in a file
(c) an operation that returns the full path of the file
(d) an operation that determines whether the file is hidden or not
(e) none of the above

20. Which one of the following symbols is used to indicate that a method is `private` in UML?

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

Part B: Terminology – worth 1 mark each

FILL IN THE SINGLE BEST ANSWER.

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

21. A(n) _____*default*_____ constructor is a no-arg constructor that is created automatically by the JVM when instantiating an object in memory, assuming such a constructor does not already exist in that class.

22. A(n) _____*aggregation*_____ relationship is a type of relationship in which one object is composed of multiple objects.

23. In Java, only _____*single*_____ inheritance is permitted; this is to avoid the diamond problem.

24. A parent class is to a _____*child*_____ class as a base class is to a _____*derived*_____ class.

Part C: Short Written Answers – marks as indicated

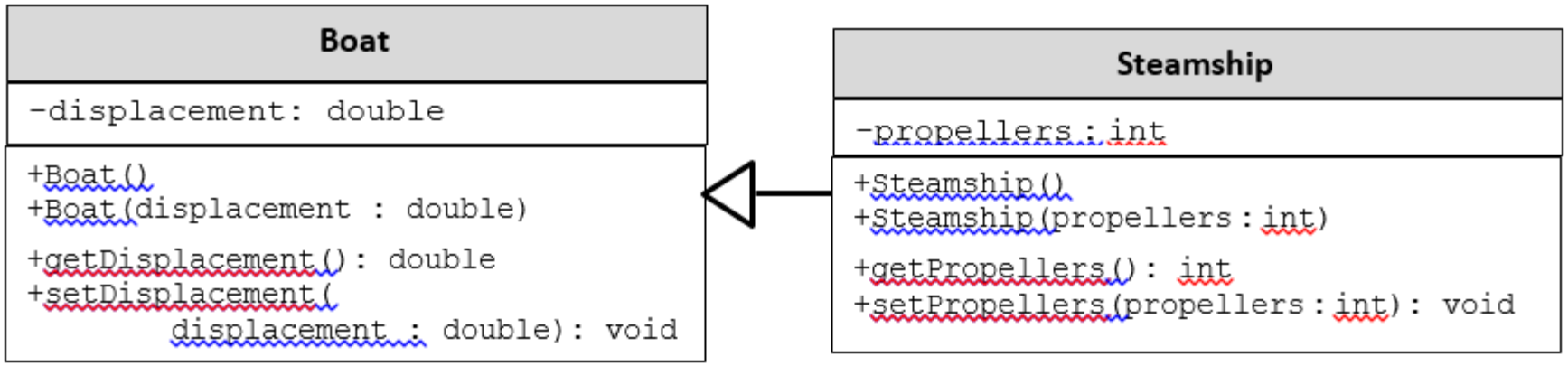
25. The `GregorianCalendar` class is a subclass of the `Calendar` class. `GregorianCalendar.YEAR` returns, as its default, the integer `1970`. Using this information, write out the full Java definition of the `YEAR` field in the `GregorianCalendar` class, including both (1) the complete declaration of the `GregorianCalendar` class itself, along with (2) the declaration of `YEAR` field within that class, assigning a default value of `1970`. (You do not need to concern yourself with any other definitions in `GregorianCalendar` other than the `YEAR` field.) **/ 6 marks**

```
public class GregorianCalendar extends Calendar {

    public static final int YEAR = 1970;

}
```

26. Consider the UML diagram below, noting that `Steamship` is a subclass of `Boat`:



Assume the code for the Boat superclass is already defined, including the two constructors indicated. In the box below, fill in the missing code for the `Steamship` subclass corresponding to the UML diagram above. Be sure to chain the no-arg `Steamship` constructor to the 1-arg constructor (assume a default of 3 propellers), and then chain to the `Boat` superclass's no-arg constructor.

/ 4 marks

```
public class Steamship   extends Boat                                    {
    private int propellers;
    public Steamship () {

        this(3);

    }

    public Steamship (int propellers) {
       super();
       setPropellers(propellers);

    }

    public int getPropellers() { return propellers; }

    public void setPropellers ( int propellers) {
        this.propellers = propellers;
    }

}
```

Part D: Spot the error – worth 6 marks total

```
public class RaceCarLauncher {
   private final static int TOTAL_CARS = 8, TRACK_LENGTH = 5000;
   private RaceCar[] raceCars = new RaceCar[TOTAL_CARS];

   public static void main(String args) {
      (new RaceCarLauncher()).loadRaceCars();
      System.out.println("The fastest Racecar is car number " + findFastestCar());
   }

   private static void compare(RaceCar c1, RaceCar c2) {
      System.out.println("Car " + c1.getCarNum() + " is " + ((c1.getKPH() >
          c2.getKPH()) ? "faster : slower") + " than " + c2.getCarNum());
   }

   private void loadRaceCars() {
      for (int carCtr = 0; carCtr < TOTAL_CARS; carCtr++) {
         raceCars[carCtr] = new RaceCar(TRACK_LENGTH, (50 * Math.random() + 120));
         raceCars[carCtr].toString();
      }
   }

   private static int findFastestCar() {
      RaceCar fastestCar = raceCars[0];
      for (RaceCars rc; raceCars)
         fastestCar = rc.getKPH() > fastestCar.getKPH() ? rc : fastestCar;
      return fastestCar.getCarNum();
   }
}
```

1 2 3 4 5 6 7 8

Part D: Spot the error – worth 6 marks total

```
public class RaceCar {
   private long distance;
   private int time, thisCarNum (9)
   private static int carNum = 1;

    public RaceCar(long distance, int time) {
      setDistance(distance); setTime(time); setCarNumberAndIncr(); (10)
    }

    public long getDistance() {return distance;}
    public void setDistance(long distance) {
      this.distance = distance;
    }
    public int getTime() {return time;}
    public void setTime(int time) {this.time = time;}

    public int getCarNum() {return thisCarNum;}
    private void setCarNumAndIncr() {thisCarNum = ++carNum;}
                                                          (11)
    public double getKPH() {
      long km = getDistance()/1000;
      double hrs = getTime()/3600); (12)
      return((double)km/hrs); (13)
    }

    @Override
    public String toString() {
      return "Car " + getCarNum() + " has speed " + String.format("%.2f", getKPH()) + " kph.";
   }
}
```

2. Array should be `static`, else won't be visible to `main()` when program loads
3. Needs () after `new RaceCarLauncher`, otherwise won't instantiate constructor
4. Should be "faster":"slower" in separate quotes, otherwise conditional if won't work.
5. Needs to be cast as int, otherwise, value passed as a double to an int
6. Needs `System.out.println()` to output String
7. Array is of type `RaceCar`, not `RaceCars`; so should be `RaceCar rc`
8. Enhanced for uses ':', not ';'
9. Missing ';' at end of line
10. Method is inconsistently named, i.e. `setCarNumberAndIncr()` called above in constructor, versus `setCarNumAndIncr()` actually declared
11. Pre-increment causes car number to be incremented before it is passed to `thisCarNum`; hence car number 1 will never exist, contrary to output
12. Since `getDistance()` and `getTime()` are integral values, these calculations will typically return 0, causing serious round-off errors in the output, or NaN; needs to be cast as a double to give accurate results
13. Extra closing parenthesis ')'; should be removed