

# Exam C – Solutions

Exam D has the same multiple choice questions, but in different order.

2. Which *one* of the following is used in UML diagrams to signal that a *member* is abstract?  
(a) underline    (b) *italics*    (c) **boldface**    (d) CAPITALIZE    (e) none of the above
3. **True** (a) or **False** (b): If a class is abstract, then all of its members must be abstract
4. Which *one* of the following methods has the same signature as `main()` in a JavaFX application (excepting the difference in the name of the method itself)?  
(a) `init()`    (b) `start()`    (c) `launch()`    (d) `stage()`    (e) none of the above
5. Which one of the following is **NOT TRUE** of method and constructor chaining?
  - (a) it is an example of code reuse
  - (b) it helps to make the client interface more intuitive by presenting a standard set of features to the user of the code
  - (c) it is less likely to generate new errors if the output needs to be altered
  - (d) it leads to faster execution times
  - (e) none of the above

6. **TRUE** (a) or **FALSE** (b): public static members in a superclass are not visible in any of its subclasses
7. **TRUE** (a) or **FALSE** (b): if new features are added to an existing program , then it is said to have been *refactored*.

Note: refactoring does not add new features to a program, it merely corrects/tidy up existing code

8. When `public static void main(String[] args)` is written in a UML class diagram, it will appear as:

- (a) `+main(args: String[]): void`
- (b) `-main(args: String[]): void`
- (c) `+main(args: String[]): void`
- (d) `+main(): void`
- (e) none of the above

9. Which *one* of the following is **NOT TRUE** of a class that contains an abstract method:

- (a) The class cannot be extended to a subclass
- (b) The class cannot be instantiated
- (c) The abstract method must be overridden in a subclass, unless the subclass is itself abstract
- (d) The class can be extended into a concrete class
- (e) None of the above

Note: either answer is correct

10. Which abstract JavaFX class is the parent superclass of all graphical JavaFX objects?  
(a) Node (b) Control (c) Pane (d) Parent (e) none of the above
11. **TRUE** (a) or **FALSE** (b): Lambda expressions can only be used with classes that contain a single method
- Note: Although this material was covered in class, it was in section 6.3 of the course notes. Since this was technically 'off-limits' for the exam, students were awarded a mark for this question regardless of the answer they actually gave
12. **TRUE** (a) or **FALSE** (b): The `new` keyword is implied whenever you make a definition like:
- ```
String month = "December";
```
13. **TRUE** (a) or **FALSE** (b): You can always extend any superclass into a subclass; there are no restrictions
14. Which *one* of the following is **NOT** one of the three basic reference types in Java?  
(a) a String (b) a class (c) an interface (d) an array (e) none of the above

15. When you make a definition like:

```
Scanner scanner = new Scanner(System.in);
```

which one of the following most correctly describes what happens after this line is executed:

- (a) scanner holds a new Scanner object
- (b) scanner holds an address that points to a new instance of a Scanner object
- ☒ (c) scanner holds a number that is tied to an address that points to a new instance of a Scanner object
- (d) scanner holds an address that points to the Scanner class loaded by the class loader
- (e) none of the above

16. Which one of the following is not part of the JRE (Java Runtime Environment)?

- (a) The JVM
- ☒ (b) .class files stored in /bin
- (c) The JRE System Library
- (d) Support/startup files
- (e) none of the above

17. **TRUE** ☒ (a) or **FALSE** (b): Whenever any new object is instantiated, Object's no-arg constructor will be loaded first, before any other constructors

18. **TRUE** (a) or **FALSE** (b): A UML diagram can describe *is a* OR *has a* relationships, but never both at the same time.
19. Which *one* of the following is used to indicate that a particular method is to be run as part of a JUnit test?

(a) `@Test`

(b) In any project, choose:

Select Project >> Generate JavaDoc

from the menu and follow the instructions

(c) `assertTrue()`

(d) `assertFalse()`

(e) none of the above

20. Which one of the following Parent objects would you instantiate and load into the scene to structure it, i.e. layout the space to allow for buttons, text, etc. to be added

(a) a Parent object

(b) a Pane object

(c) a Control object

(d) a Node object

(e) none of the above

21. Which one of the following is **TRUE** of the role that the EventHandler object plays in dealing with user generated events, such a mouse click?
- (a) it connects the event that happened with the code to be executed as a result
  - ☒ (b) it contains an abstract method that must be overridden by the class that implements it
  - (c) it encodes the event that occurred, which is passed to the code to be executed
  - (d) it is an abstract class that represents the node on which the event occurred
  - (e) none of the above
22. **TRUE** ☒ (a) or **FALSE** (b): the reference value of an object is passed to a method via the stack, rather than the heap
23. **TRUE** (a) or **FALSE** ☒ (b): You cannot both *override* and *overload* a class's methods

**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**

24. final classes cannot be extended, while final methods cannot be overridden.

25. 'has a' relationships are of two types: composition and aggregation.

25(alt). API stands for Application Program interface

26. When an inner class is not instantiated first into a named object, but instead is passed directly into a method without an identifier, it is referred to as an anonymous inner class.

26(alt) When one class is located inside another, it is referred to as a(n) inner class

27. After an object has been instantiated, setting its value to null will invoke garbage collection

28. You can load a Button object into a Scene because a Button 'is a' Parent object. This feature is due to one of the four pillars of OOP, known as polymorphism.

## Part C: Written Answers

### Part C: Short Written Answers – ANSWER ANY 2 QUESTIONS OUT OF THE FOLLOWING 3: ONLY THE FIRST 2 QUESTIONS ANSWERED WILL BE MARKED

**29.** `System.in` is used by the `Scanner` class to indicate that input will be taken from the keyboard. Inside the `System` class, `in` stores a value of type `InputStream`. Given this information, along with information conveyed by the format of the `System.in` object itself, write, in the space below (1) the declaration of the `System` class, and (2) the declaration of the `in` variable, as you would expect to see it written in inside that class.

```
public final class System {  
    public static InputStream in;  
}
```

Note: no marks lost for missing 'final' in the class header, since, while this is in the notes, it isn't apparent from `System.in`. However, everything else is.

## Part C: Written Answers

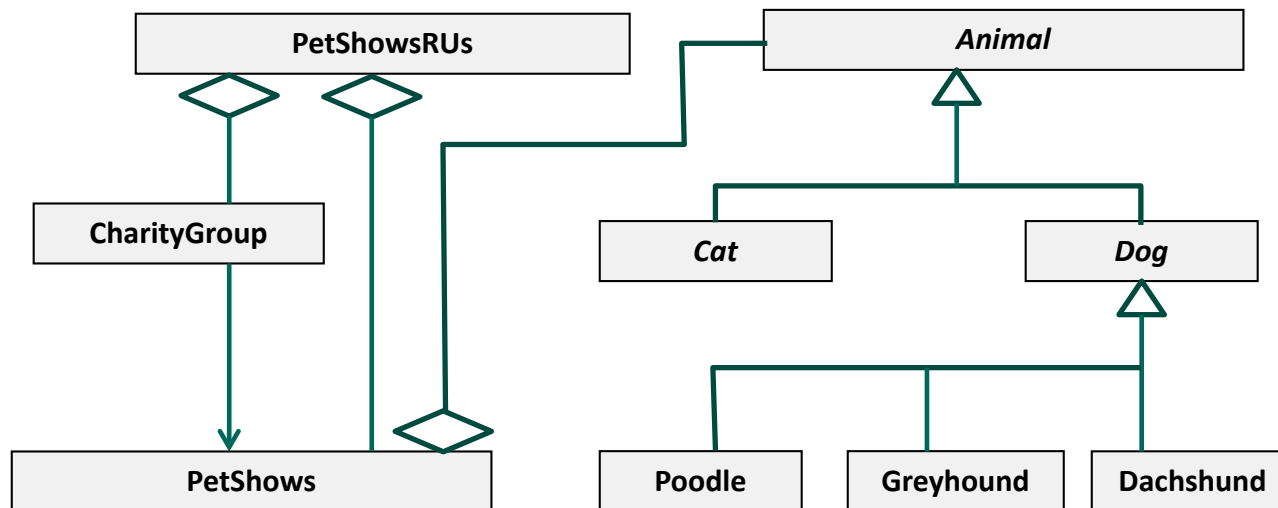
### Part C: Short Written Answers – ANSWER ANY 2 QUESTIONS OUT OF THE FOLLOWING 3: ONLY THE FIRST 2 QUESTIONS ANSWERED WILL BE MARKED

**30.** Can an abstract method be declared in a final class? Explain why or why not in one concise sentence in the space provided below

*A final class **cannot** be overridden, but any class that contains an abstract method must be abstract itself, and hence **must** be overridden to be of any use (since it must be instantiated to an object at some point.)*

**31.** A company called PetShowsRUs helps charity groups organize one-time pet shows featuring specialized breeds of dogs and cats. The software company you work for is required to write Java code that models the relationship between PetShowsRUs, its charities, the PetShows it produces, and the animals it features. Given the following UML classes (and *only* the classes provided), use proper UML notation to indicate the relationships between these classes (i.e. as aggregation, composition, inheritance and association). (Note: Poodles, Greyhounds, and Dachshunds are special breeds of dog.)

Answer: See below. Note that other variations were allowed for the relationship between PetShowRUs, CharityGroup, and PetShows, sometimes with a slight reduction in marks



**32.** The code on the next three pages, for which the description is given below, contains no fewer than 16 errors, which could be manifested as either compile-time or run-time errors. The latter may be due to logic errors on the part of the programmer. Your job is to find 12 of those errors. Place a number from 1 to 12 next to each of these errors, circle each of these numbers, and explain why the code is in error in the box on page 7, next to the number you just circled in the code.

**Description:** The following program\* consists of five classes, each shown in its own box. A **Shape** class stores the `x` and `y` coordinates of a Shape object. It contains an abstract method `calcArea()`. Additionally, it contains the `inputAllFields()` method that prompts the user to input the `x` and `y` coordinates. **Circle** and **Rectangle** are two classes that extend Shape. They each override `calcArea()` with an appropriate calculation. Additionally, they override `inputAllFields()` and `toString()` in the superclass and append their own information on to these methods. The **Input** class contains two static methods that prompt the user for input, display a string output, and then checks to see that the input is between two appropriate ranges before prompting the user for input; otherwise, the user is prompted to re-enter the values. Finally, the **TestShapes** class runs the `main()` routine, which prompts the user to input some number of shapes, and prompts the user for the coordinates and measurements of each shape, before finally calculating each result. Note that for simplicity, the only shapes used are `Circles` and `Rectangles`.

**32. (con't)** *Typical* output is shown below right. This is given as a guide only; it should NOT be assumed to be character-by-character identical to actual output. (i.e Marks are not given for apparent missing or added spaces or lines based on this sample)

```
Input number of shapes (20 max):3
```

```
Choose 1:Circle 2:Rectangle(1-2)1
```

```
Enter x:(0.0-1920.0)100
```

```
Enter y:(0.0-1080.0)200
```

```
Enter radius:(1.0-1000.0)30
```

```
Choose 1:Circle 2:Rectangle(1-2)2
```

```
Enter x:(0.0-1920.0)52.2
```

```
Enter y:(0.0-1080.0)49.3
```

```
Enter length:(1.0-1000.0)30
```

```
Enter width:(1.0-1000.0)6
```

```
Choose 1:Circle 2:Rectangle(1-2)2
```

```
Enter x:(0.0-1920.0)92.3
```

```
Enter y:(0.0-1080.0)44.3
```

```
Enter length:(1.0-1000.0)4
```

```
Enter width:(1.0-1000.0)5
```

```
Calculating Area of Shapes
```

```
Circle coordinates are (100.0, 200.0)
```

```
area = 2827.43 radius = 30
```

```
Rectangle coordinates are (52.2:49.3)
```

```
area = 180.00 length: 30.0 width: 6.0
```

```
Rectangle coordinates are (92.3:44.3)
```

```
area = 20.00 length: 4.0 width: 5.0
```

\*From code originally supplied by Rex Woollard, from his Fall 2015 midterm. Note that in the original version of this problem, students were expected to write four out of the five classes shown here. So if you think *this* exam is tough, imagine having to write most of this code during a midterm.

```
public class TestShapes {
```

```
    public static void main(String[] args) {  
        private Shape[] listShapes = new Shape[20];
```

① Should be 20, not 10, given the array declaration and the description

```
        int numShapes = Input.getInt("Input number of shapes (20 max):", 1, 10);  
        for (int i = 0; i < numShapes; i++) {
```

```
            Shape shape;
```

```
            if (Input.getInt("\nChoose 1:Circle 2:Rectangle", 1, 2) == 1)
```

```
                ② circle = new Circle();
```

```
            else
```

```
                ③ shape = new Rectangle();
```

```
                shape.InputAllFields();
```

```
                listShapes[i] = shape;
```

```
        }
```

② Should be shape = new Circle();

③ should be inputAllFields();

④ should be listShapes

```
        System.out.print("\nCalculating Area of Shapes\n");
```

```
        for (Shape shape : Shape[]); ④ ⑤
```

```
            System.out.println(shape);
```

⑤ for loop should not end in ;

```
    }
```

⑥ Missing closing '}'

⑥

```

7
public class Shape {
    private static final double MAX_X = 1920.0, MAX_Y = 1080.0;
    private double x, y;

    public Shape() {}
    public abstract double calcArea();
    public void inputAllFields() {
        x = Input.getDouble("Enter x:", 0.0, MAX_X);
        y = Input.getDouble("Enter y:", 0.0, MAX_Y);
    }

    @Override
    public String toString() {
8
        return((this instanceof Circle)?"Circle":"Rectangle" +
            " coordinates are (" + x + ", " + y + "), area = " + calcArea());
    }
}

```

7 class contains abstract method, hence must be abstract itself

8 this always refers to Shape, and Shape is never an instance of a Circle. Hence this statement always returns "Rectangle", never "Circle"

```

public class Circle extends Shape {
    private final double MIN_RADIUS = 1.0;
    private final double MAX_RADIUS = 1000;
    private double radius;

    @Override
    public void inputAllFields() {
        super.inputAllFields();
        radius = Input.getDouble("Enter radius:",
                                MIN_RADIUS, MAX_RADIUS);
    }

    public double calArea() {
        return Math.Pi * radius;
    }

    @Override
    public String toString() {
        return (super.toString() + "radius: " + radius);
    }
}

```

9 Should be calcArea not calArea, to be consistent with superclass method

10 Should be Math.PI, not Math.Pi

```

public class Rectangle extends Shape {
    private final double MIN_DIM = 1.0;
    private final double MAX_DIM = 1000;
    private double length, width;

    @Override
    public void inputAllFields() {
        super.inputAllFields();
        length = Input.getDouble("Enter " +
                                "length:", MIN_DIM, MAX_DIM);
        width = Input.getDouble("Enter " 11
                                "width:", MIN_DIM, MAX_DIM);
    }

    public double calcArea() {
        return getlength() * getwidth(); 12
    }

    public String toString() {
        return (super.toString() + " length: "
            + length + " width: " + width);
    }
}

```

11 Missing '+' to connect the two Strings together

12 getLength() and getWidth() are not declared anywhere

```

13 public static class Input {
14     private Scanner input = new Scanner(System.in);
15     private static int d; private static int i;

```

13 Missing import Scanner

```

16 public static double getDouble(String s, double min, double max){
    System.out.print(s + "(" + min + "-" + max + ")");
    do{
        d = input.nextDouble();
        input.nextLine();
    } while ((d < min) || (d > max));
    return d;
}

```

14 No such thing as a static class in Java, i.e. you can't make everything inside by adding static to the class header

```

public static int getInt(String s, int min, int max){
    System.out.print(s + "(" + min + "-" + max + ")");
    do{
        i =input.nextInt();
        input.nextLine();
    } while ((i < min) || (i > max))17
    return i;
}
}

```

15 Scanner must be static

16 Should be double d, not int d

17 Missing semicolon at end of while loop

18. Actual output displays "(1-20)", but code sample says "(20 max)"

## Things which are *NOT* errors, but which are frequently listed as such by students:

| Suspected Error                                                                      | Reason typically given                                                                        | Reason why this is not an error                                                                                       |
|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <code>private Shape[] listShapes<br/>= new Shape[20];</code>                         | Can't instantiate Shape, since it must be an abstract class                                   | You're not instantiating a Shape object, you're instantiating an array of type Shape                                  |
| <code>Shape shape;</code>                                                            | Needs <code>= new Shape();</code> to work properly                                            | This is a declaration only; the concrete subclass (Circle/Rectangle) is assigned in the code that follows.            |
| <code>Shape shape;</code>                                                            | Inside <code>for</code> loop; Shape therefore declared repeatedly                             | Shape could be declared outside the <code>for</code> , but there is not error putting it inside.                      |
| Missing Scanner                                                                      |                                                                                               | You don't use Scanner until the last class; therefore no Scanner needed elsewhere                                     |
| <code>@Override<br/>public String toString() {<br/>return((this instanceof...</code> | Shouldn't use @Override, since nothing to override                                            | <code>toString()</code> exists in Object superclass, hence you can/should override it.                                |
| <code>this instanceof Circle</code>                                                  | Should be <code>instanceOf</code> , with capital 'O'                                          | <code>instanceof</code> is an operator, not a method, hence it doesn't follow Java naming conventions                 |
| <code>((this instanceof ...)?<br/>"Circle":"Rectangle" + ...</code>                  | Not a correct statement;<br>? should not be in statement;<br>+ cannot be added to end... etc. | This is normal usage for a ternary/conditional for loop (If you're still struggling with this, time to deal with it.) |

## Things which are *NOT* errors, but which are frequently listed as such by students:

| Suspected Error                                       | Reason typically given                                                            | Reason why this is not an error                                                                                                                                                                                                        |
|-------------------------------------------------------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>super.inputAllFields();</code>                  | Shouldn't call <code>super.inputAllFields()</code>                                | Need call to superclass method in order to enter and save x and y coordinates                                                                                                                                                          |
| <code>private final double<br/>MAX_DIM = 1000;</code> | Should be 1000.0, not 1000, since MAX_DIM is a double                             | Integer is upcast to double automatically, hence no error loading MAX_DIM                                                                                                                                                              |
| <code>@Override</code>                                | Must use override before <code>toString()</code> , <code>calcArea()</code> , etc. | Not an error if the subclass signature matches the superclass.                                                                                                                                                                         |
| Must use { } inside if / else and for statements      |                                                                                   | No! No! No! You don't need the braces if there is <i>only a single operation involved</i>                                                                                                                                              |
| <code>System.out.println(shape);</code>               | Can't print out shape object                                                      | When you attempt to print an object, you call the object's <code>toString()</code> method by default. If this is overloaded (as it is in the code provided) the overloaded <code>toString()</code> gets used to provide String output. |
| <code>public abstract<br/>double calcArea();</code>   | Shouldn't be abstract                                                             | Of course it should! You can't calculate the area of a Shape until you've subclassed it into <i>some kind of</i> Shape.                                                                                                                |