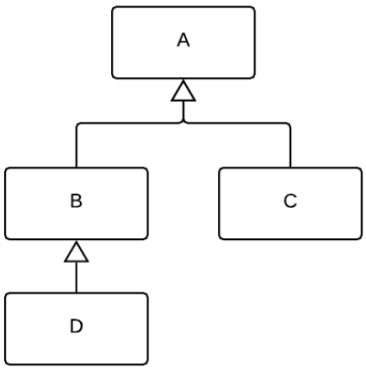
Started on	Wednesday, 5 April 2023, 12:38 PM
State	Finished
Completed on	Wednesday, 5 April 2023, 1:57 PM
Time taken	1 hour 19 mins

Question **1**

Complete

Marked out of 1.00

Given the following UML diagram:



Which of the following represents an **ILLEGAL** cast?

a. more than one of these

b. A a = (A) new D();

c. B b = (B) new D();

d. A a = (B) new D();

e. B b = (B) new C();

Question **2**

Complete

Marked out of 1.00

In an application that uses **unchecked** exceptions, should all exception possibilities be caught and handled?

- a. No. You need only catch and handle amy unchecked exceptions you want to prevent from unexpectedly crashing the program
- b. Yes. A program can not attempt any input/output otherwise
- c. Yes. If not all unchecked exceptions are caught and handled or delegated, the program will not compile
- d. No. However, all unchecked exceptions from the same family (class hierarchy) should always be caught and handled

Question **3**

Complete

Marked out of 6.00

Match the following Java AWT/Swing components and features with what they are each used for.

** Please note this question is worth 6 marks

requestFocus()	a method used to achieve absolute positioning of a JComponent
JTextField	Provides a space for the user to type into
MouseEvent	is an event generated whenever a mouse is clicked or moved
Timer	an object used to emit ActionEvents at regular periodic intervals, and route them to a specified listener to achieve automate
JComponent	All Swing components inherit from this class.
setBounds(..)	a method to re-route events back toward an existing JComponent that is registered to a relevant KeyListener

Question **4**

Complete

Marked out of 1.00

What is the default layout manager for a **JFrame** ?

- a. BorderLayout
- b. FlowLayout
- c. GridLayout
- d. BoxLayout
- e. GridBagLayout

Question **5**

Complete

Marked out of 1.00

What constraints are there when initializing new objects using the **super** reference as a method call?

- a. It must occur as the first statement of any constructor
- b. It can only be used by constructors defined in the parent
- c. There are no restrictions on the use of **super**
- d. Only one child classes that override the default constructor can use it
- e. It can only be used at the end of a constructor

Question **6**

Complete

Marked out of 1.00

Which of the following would be considered a benefit of **polymorphism**?

- a. Polymorphism allows for multiple parts of a program to be run concurrently
- b. Variables can be re-used in order to save memory
- c. Constructing new objects from old objects of a similar type saves time
- d. The same program logic can be used with objects of several related types.

Question **7**

Complete

Marked out of 1.00

What is the **primary** purpose of following an **M-V-C** architecture?

- a. A way of restricting an interactive application to only use sychronous (non user driven inputs/events)
- b. A mechanism for modeling a set of classes that can efficiently represent the backend data/state (model) of an interactive application (without concern for its visual representation).
- c. To decouple any manipulations of the state/data relating to an application, from the visual presentation of content, and the interactions/interfacing performed by the end user
- d. a way of merging together (into a single class), the gui application and any relevant data needed to represent the state of the application

Question **8**

Complete

Marked out of 1.00

Consider the following class that represents a line segment connecting two points (its start point and its end point):

```
public class LineSegment {

    private final Point2 start;
    private final Point2 end;

    public LineSegment(Point2 p1, Point2 p2) {

        this.start = p1;
        this.end = p2;

    }

    // remainder of class not shown
}
```

Which statement best describes the class **LineSegment**?

- a. **LineSegment** is a superclass of the class **Point2**
- b. **LineSegment** is an aggregation of two points
- c. **LineSegment** is a composition of two points
- d. **LineSegment** is an association that is neither aggregation nor composition
- e. **LineSegment** is a subclass of the class **Point2**

Question **9**

Complete

Marked out of 1.00

Can an interface name be used as the type of a variable, like this:

```
public static void main( String[] args ) {
```

```
        SomeInterface x;
         //...

}
```

- a. No: a variable must always be an object reference type or a primitive type
- b. Yes: the variable can refer to any object who's class inherits from the interface
- c. No: a variable must always be an object reference type
- d. Yes: the variable can refer to any object who's class implements the interface
- e. No: a variable must always be a primitive type

Question **10**

Complete

Marked out of 1.00

Consider the following class **FreshFood**.

```
public class FreshFood {

    private String name;
    private int age; // in days
    private int expiresIn; // in days

    public FreshFood(String name, int days) {
        System.out.println("mmm");
        this.name = name;
        this.age = 0;
        this.expiresIn = days;
    }

    public FreshFood(FreshFood f) {
        System.out.println("yum");
        this.name = f.name;
        this.age = f.age;
        this.expiresIn = f.expiresIn;
    }

    public boolean isExpired() {
        return (this.expiresIn < this.age);
    }

    public void age(int days) {
        this.age += days;
    }

    public int daysLeft() {
        return (this.expiresIn - this.age);
    }

}
```

The class **FreshFood** contains:

- a. Two copy constructors
- b. A custom and a copy constructor
- c. A default and a custom constructor
- d. Two default constructors
- e. A default and a copy constructor
- f. Two custom constructors

Question **11**

Complete

Marked out of 1.00

Given the following two classes (related by inheritance):

```
public class Reptile {

   private int size;    // Assume size>0 && size<=10

   public Reptile (int size) { /* IMPLEMENTATION NOT SHOWN */ }

   // Sets the size of this reptile to be equal to size
   public void setSize(int size) { /* IMPLEMENTATION NOT SHOWN */
   }

}

public class Lizard extends Reptile {

   public Lizard(int size) {  /* MISSING (30) */ }

   // Sets the size of this lizard to be equal to size
   @Override
   public void setSize(int size) {  /* MISSING (31) */  }

}
```

What can the line with the comment **/* MISSING (30) */** be replaced with in the Lizard constructor (select ALL that apply) ?

- [ ] a. `super.setSize(size);`
- [x] b. `super(size);`
- [ ] c. `this(size);`
- [ ] d. `this.size = size;`
- [ ] e. `this.setSize(size);`

Question **12**

Complete

Marked out of 1.00

Assume that a user wishes to create a rectangular selection tool, where they click on a location in a GUI window, hold the mouse button and release when the mouse is at a different location (using this to dynamically draw a rectangle to represent the selection area).

Which listener(s) would need to be implemented to achieve this?

** this question is worth 2 marks

- a. MouseMotionListener only
- b. MouseMotionListener and MouseListener
- c. MouseListener only
- d. MouseMotionListener and KeyListener
- e. ActionListener only

Question **13**

Complete

Marked out of 1.00

Consider the classes defined in the following code fragment:

```
public class Marsupial {
     /* no constructors or other methods have been declared */
}

public class Kangaroo extends Marsupial {
     /* no constructors or other methods have been declared */
}

public class BigRed extends Kangaroo {
     /* no constructors or other methods have been declared */
}
```

Which of the following object declarations will **NOT** compile (select ALL that apply)?

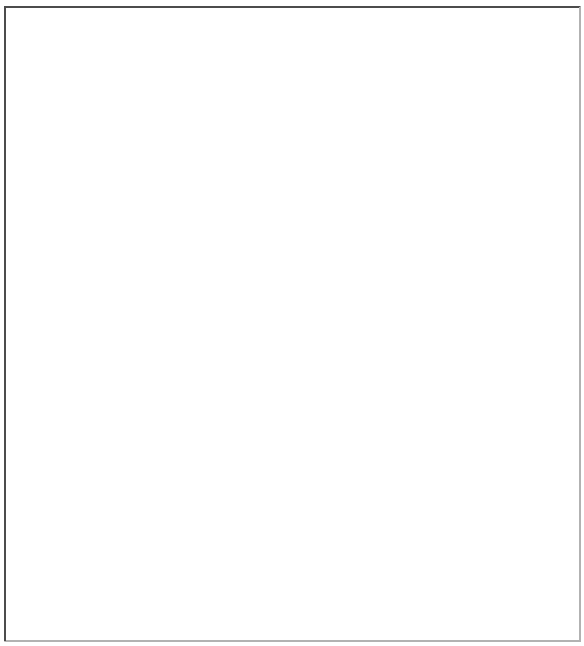
- a. BigRed anim = new Kangaroo();
- b. `Kangaroo anim = new BigRed();`
- c. `BigRed anim = new Marsupial();`
- d. `Marsupial anim = new Kangaroo();`

Question **14**

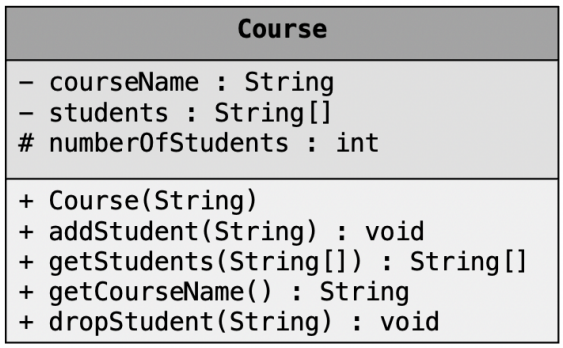
Complete

Marked out of 1.00

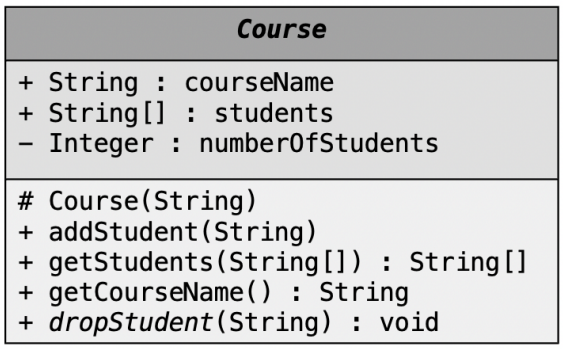
Which of the following UML diagram best represents the class **"Course"** (as defined below)?



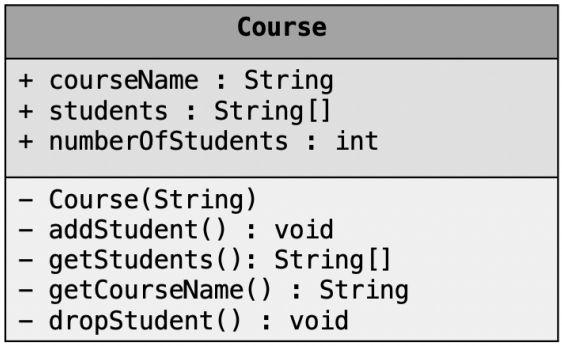
a.



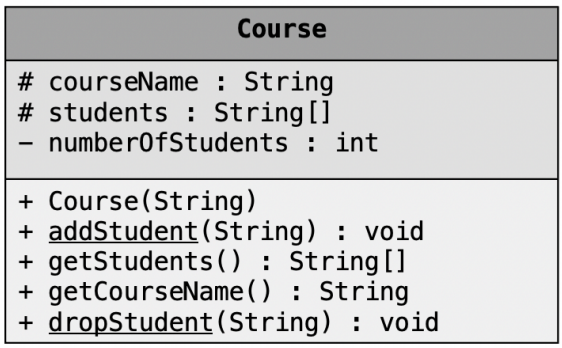
b.



c.


Course
+ courseName : String
+ students : String[]
+ numberOfStudents : int
- Course(String)
- addStudent() : void
- getStudents(): String[]
- getCourseName() : String
- dropStudent() : void

d.


Course
courseName : String
students : String[]
- numberOfStudents : int
+ Course(String)
+ addStudent(String) : void
+ getStudents() : String[]
+ getCourseName() : String
+ dropStudent(String) : void

Question **15**

Complete

Marked out of 1.00

Given the following two classes (related by inheritance):

```
public class Reptile {

   protected int size;    // Assume size>0 && size<=10

   public Reptile (int size) { /* IMPLEMENTATION NOT SHOWN */ }

   // Sets the size of this reptile to be equal to size
   public void setSize(int size) { /* IMPLEMENTATION NOT SHOWN */
   }

}

public class Lizard extends Reptile {

   public Lizard(int size) {  /* MISSING (30) */ }

   // Sets the size of this lizard to be equal to size
   @Override
   public void setSize(int size) {  /* MISSING (31) */  }

}
```

What can the line with the comment **/* MISSING (31) */** be replaced with (select ALL that apply) ?

- [ ] a. `super.setSize(size);`
- [ ] b. `this.setSize(size);`
- [ ] c. `this(size);`
- [ ] d. `super(size);`
- [x] e. `this.size = size;`

Question **16**

Complete

Marked out of 1.00

The main reason you would use an *inner class* within a Java Swing GUI application is so that you can:

a. Implement a single event handler to handle all possible action events

b. Implement more than one event handler for handling action events

c. Inherit state and behaviour from another (external) class

d. Have direct access to any private instance variables from the "outer" (main) class

e. Keep your Java class more organized

Question **17**

Complete

Marked out of 1.00

Given the following code fragment:

```
// assume Citizen is the parent class of Voter
// assume Voter is the parent class of RegisteredVoter
// and that both Voter and RegisteredVoter classes have defined the method castVote()

Voter person;
person = new RegisteredVoter("Bob", 42);
person.castVote();
```

When the **castVote()** method is run in the last statement, **which version of castVote() is actually run**?

a. The one defined for **Voter** because that is the type of the variable **person**

b. Both will be run (**Voter**'s version first, then **RegisteredVoter**'s version second)

c. The one defined for **RegisteredVoter** because that is the actual type of the object referred to by **person**

d. Neither; the assignment statement will cause an error/exception

e. The one closest in the source code to the **person.castVote()** statement

Question **18**

Complete

Marked out of 1.00

Which of the following statements is FALSE?

a. A single **catch(..){ }** block can be used to catch more than one type of exception

b. Some statements inserted within a **try{ }** block may never throw an exception

c. A **try{ }** block must always come immediately before any **catch{ }** block(s).

d. The statements inside a **try{ }** block cannot include any form of branching

e. It is possible for several different exceptions to be thrown from a given **try{ }** block

Question **19**

Complete

Marked out of 1.00

Which of these methods can be used to know which key(s) have been pressed in a Swing application? (Select ALL that APPLY)

a. getModifier()

b. getKey()

c. getKeyCode()

d. getActionKey()

e. getActionEvent()

Question **20**

Complete

Marked out of 1.00

Which of the following is true?

a. A subclass class can **extend** just one parent and can **implement** zero or more interfaces

b. A subclass class can **extend** zero or more parents, and can **implement** zero or more interfaces

c. A subclass class can **extend** a parent or **implement** an interface, but not both

d. A subclass class can **extend** just one parent and can **implement** just one interface

Question **21**

Complete

Marked out of 1.00

A **wrapper** class is:

- a. a class that **embeds** another class within its body
- b. a reference type (class) that **aggregates** one or more other classes
- c. a reference type (class) that **encapsulates** one of the primitive types
- d. a reference type (class) that may **not** be instantiated
- e. a reference type (class) that is **composed** of one or more other classes

Question **22**

Complete

Marked out of 1.00

Consider the following fragment of code:

```
// Assume Fruit is the superclass of Banana
```

```
Fruit myLunch = new Banana("Organic");
Object obj = myLunch;
boolean thing = obj instanceof Banana;
```

What value is most likely assigned to thing?

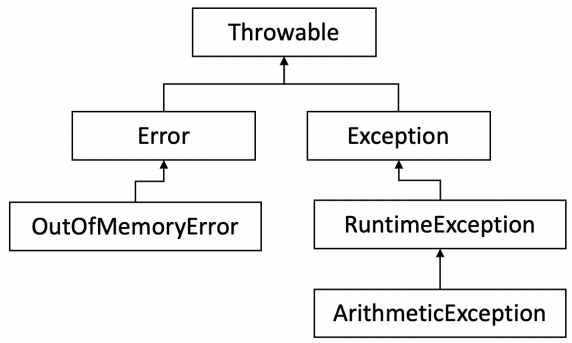
- a. `false`
- b. `true`
- c. `"Organic"`
- d. nothing, there will be a runtime error
- e. `null`

Question **23**

Complete

Marked out of 1.00

Given the following class hierarchy:



If a **try { ... }** block could potentially throw **any** of the following objects: **Throwable**, **Error** and **OutOfMemoryError**, what order would we have to list individual **catch(...) { ... }** blocks after the **try { ... },** in order to make sure each object can be handled in its own unique way?

- a. Throwable, Error, OutOfMemoryError
- b. Error, Throwable, OutOfMemoryError
- c. OutOfMemoryError, Error, Throwable
- d. Error, OutOfMemoryError, Throwable

Question **24**

Complete

Marked out of 1.00

What is it called when a program is written to respond to the mouse/button clicks, key presses, menu selections, and other user-based interactions within a given Java application?

- a. User-driven programming
- b. Exception-driven programming
- c. Action-driven programming
- d. Peripheral-driven programming
- e. Event-driven programming

Qu
25

Co

Ma
ou
of
1.0

Qu
2

Co

Ma
ou
of
1.0

Qu
2

Co

Ma
ou
of
1.0

Qu
2

Co

Ma
ou
of
1.0

Qu
2

Co

Ma
ou
of
1.0

Qu
3
Co
Ma
ou
of
1.0