|  |
| --- |
| **Exercise 3.1**  The class diagram contains only 2 elements: LabClass and Student. The LabClass class is linked to the Student class.  The object diagram contains 4 elements. 1 LabClass object and 3 Student objects. The LabClass object contains links to the three Student objects. |
| **Exercise 3.2**  A class diagram changes when you modify the source code. That can be by changing the relations between classes or creating/deleting classes. |
| **Exercise 3.3**  An object diagram changes when the program is running. It can be changed by creating new objects, calling methods, and making assignments involving object references. |
| **Exercise 3.4**  private Instructor tutor; |
| **Exercise 3.6**  **You should call getValue and if it returns 0 you should call increment on the hours object.** |
| **Exercise 3.10**  **Error: non-static method getValue() cannot be referenced from a static context** |
| **Exercise 3.11**  **Error: '.class' expected** |
| **Exercise 3.15**  false true false false true |
| **Exercise 3.16**  **The long version of this would be:**  (a == true && b == true) || (a == false && b == false)  This can be simplified to:  (a && b) || (!a && !b)  But since both must have identical values, the simplest form is:  a==b |
| **Exercise 3.17**  As in the previous exercise, the long version would be:  (a == true && b == false) || (a == false && b == true)  This can be simplied to:  (a && !b) || (!a && b)  Or:  (a || b) && (a != b)  Or even:  a != b |
| **Exercise 3.18**  No.  The method assumes that the value will only contain two digits. |
| **Exercise 3.19**  No. |
| **Exercise 3.20**  **“12cat” and “cat39”** |
| **Exercise 3.23**  It needs 60 clicks Using the method setTime() on the object. |
| **Exercise 3.24**  NumberDisplay digit = new NumberDisplay(10); |
| **Exercise 3.25**  Rectangle window = new Rectangle(3,6); |
| **Exercise 3.26**  It initializes the time to the values passed to the method. It uses the method setTime to set the time to the initial value. |
| **Exercise 3.27**  Both constructors creates two new NumberDisplays. The first constructor calls updateDisplay and the second calls setTime(hour, minute). In the second constructor there is no call to updateDisplay because this will be done in the method setTime. |
| **Exercise 3.28**  p1.print("file1.txt", true);  p1.print("file2.txt", false);  int status;  status = p1.getStatus(12);  status = p1.getStatus(34); |
| **Exercise 3.29**  **Two Square objects, a Triangle and a Circle.** |
| **Exercise 3.30**  **changeSize, moveHorizontal, moveVertical and makeVisible.** |
| **Exercise 3.31**  **No.** |
| **Exercise 3.32**  2 |
| **Exercise 3.34**  -4, -3, -2, -1, 0, 1, 2, 3, 4 |
| **Exercise 3.35**  Values in the range from -(m-1) to (m-1) |
| **Exercise 3.36 and 3.37**  1) change the updateDisplay in ClockDisplay as this:  /\*\*  \* Update the internal string that represents the display.  \*/  private void updateDisplay()  {  int hour = hours.getValue();  String suffix = "am";  if(hour >= 12) {  hour = hour - 12;  suffix = "pm";  }  if(hour == 0) {  hour = 12;  }  displayString = hour + "." + minutes.getDisplayValue() + suffix;  }  2)  public ClockDisplay()  {  hours = new NumberDisplay(12); //changed  minutes = new NumberDisplay(60);  updateDisplay();  }  public ClockDisplay(int hour, int minute)  {  hours = new NumberDisplay(12); //changed  minutes = new NumberDisplay(60);  setTime(hour, minute);  }  private void updateDisplay()  {  int hour = hours.getValue();  if(hour == 0) {  hour = 12;  }  displayString = hour + "." + minutes.getDisplayValue();  } |
| **Exercise 3.41**  **An example of the output of showNotes:**  Instrument number: 0  C (12) C (12) D (12) C (12) D (12) E (12) C (12) |
| **Exercise 3.43**  **Yes, the tune will be played again each time it is passed to playTune.** |
| **Exercise 3.45**  **The following is another internal method call:**  playTrack(sequence)  **The following are all examples of external method calls in the playTune method:**  tune.getNotes()  sequence.createTrack()  track.add(...)  note.getNoteName()  note.getDuration() |
| **Exercise 3.51**  /\*\*  \* Constructor for ClockDisplay objects.  \* This constructor creates a new clock set  \* to the current time.  \*/  public ClockDisplay()  {  hours = new NumberDisplay(24);  minutes = new NumberDisplay(60);  long millis = System.currentTimeMillis();  long seconds = millis / 1000;  int currentMinutes = (int) (seconds / 60 % 60);  int currentHours = (int) (seconds / (60 \* 60) % 24);  hours.setValue(currentHours);  minutes.setValue(currentMinutes);  updateDisplay();  } |
| **Exercise 3.52**  **It does give the same results as the previous version.** |
| **Exercise 3.61**  **A third NumberDisplay object would need to be created in both constructors. The setTime method would add a third parameter. The updateDisplay method would need to build the displayString from the three values. The logic of the timeTick method becomes more complicated. The seconds must be incremented and if they roll over then the minutes must be incremented. The hours are incremented as currently.** |
| **Exercise 3.62**  **public void timeTick()**  **{**  **seconds.increment();**  **if(seconds.getValue() == 0) {**  **// it just rolled over!**  **minutes.increment();**  **if(minutes.getValue() == 0) {**  **hours.increment();**  **}**  **}**  **updateDisplay();**  **}** |
| **Exercise 3.63**  **In theory it could, but the logic would be unwieldy.** |
| **Exercise 3.64**  **This is likely to be very challenging at this stage. The NumberDisplay class would have an additional field such as:**  **private NumberDisplay nextTimeUnit;**  **This would need be set from a parameter passed in by the ClockDisplay. For instance,**  **public ClockDisplay()**  **{**  **hours = new NumberDisplay(24, null);**  **minutes = new NumberDisplay(60, hours);**  **seconds = new NumberDisplay(60, minutes);**  **}**  **The increment method in NumberDisplay would be:**  **public void increment()**  **{**  **value = (value + 1) % limit;**  **if(value == 0) {**  **if(nextTimeUnit != null) {**  **nextTimeUnit.increment();**  **}**  **}**  **}** |
| **Exercise 3.65 and 3.66**  **public class Tree**  **{  // The tree’s trunk.**  **private Square trunk;**  **// The tree’s leaves.**  **private Triangle leaves;**  **/\*\***  **\* Constructor for objects of class Tree**  **\*/**  **public Tree()**  **{**  **trunk = new Square();**  **leaves = new Triangle();**  **setup();**  **}**  **/\*\***  **\* Set up the trunk and the leaves of the tree.**  **\*/**  **public void setup()**  **{**  **leaves.changeSize(120, 140);**  **leaves.moveVertical(-140);**  **leaves.moveHorizontal(128);**  **trunk.makeVisible();**  **leaves.makeVisible();**  **}**  **}** |