# CIS 162 Project 2

# A Concert

# Due Date(s)

At the beginning of lab on week 8 – See due date on Blackboard for your specific section.

Suggested phases:

* Phase 1 – Thursday, October 3, 2019
* Phase 2 – Tuesday, October 8, 2019
* Phase 3 – Beginning of lab on week 8 - See due date on Blackboard for your specific section.

# Before Starting the Project

* Read chapter 5, chapter 6 and chapter 7 (sections 7.1 – 7.4)
* Read this entire project description before starting

# Learning Objectives

After completing this project, you should be able to:

* *write* a complete Java class
* *write* methods to meet specific requirements
* *write* conditional statements with Boolean expressions
* *write* loops
* *explain* the differences between local variables, instance variables (class fields) and method parameters

# Project Summary

You will create a class to simulate the basic functionality of a concert. It will contain basic information about the concert including the date, the artist and the venue. It will allow customers to buy tickets and it will allow the concert staff to print a summary of the tickets sold for the concert.

# Class Specifications

The design specifications of a class include:

1. declarations for a list of instance fields (think of these as nouns) and
2. method declarations (think of these as verbs), including parameters, which determine how objects are used.

Implement a class, called **Concert**, with the following instance fields and methods. **Exact spelling is required** for the name of the Concert class and the headers of all the methods. Do not create additional methods or make any changes to the following requirements. If you do so, the automated tests will likely fail.

**Class Fields/Instance Variables**

Declare meaningful names with appropriate data types for each of these private instance variables:

* three integers for the *date of the concert (day, month and year)*
* a String for the *artist name*
* a String for the *venue of the concert*
* an integer for the *number of tickets available in the upper section*
* an integer for the *number of tickets available in lower section*
* an integer for the *number of tickets available in floor section*
* a double for the *concert total sales*
* final doubles for the price of the different types of tickets and the total number of tickets for each section.

/\*\* ticket prices for the different sections \*/

private static final double PRICE\_UPPER\_TICKET = 29.90;

private static final double PRICE\_LOWER\_TICKET = 99.0;

private static final double PRICE\_FLOOR\_TICKET = 180.0;

/\*\* total number of tickets per section \*/

private static final int TOTAL\_NUMBER\_UPPER\_TICKETS= 300;

private static final int TOTAL\_NUMBER\_LOWER\_TICKETS= 300;

private static final int TOTAL\_NUMBER\_FLOOR\_TICKETS= 400;

# Phase 1 (20 pts)

**Constructors**

A *constructor* is a special method with the same name as the class and generally initializes the instance variables to appropriate starting values.

* public Concert () –default constructor that assigns the following initial values to the instance variables:
  + concert date: month (9), day (8), year (2019)
  + artist: Jonas Brothers
  + venue: Van Andel Arena
  + sets the total sales to zero
  + sets the number of available tickets for the different sections to the corresponding constant variable that contains the total number of tickets for that section.

Example:

availableUpperTickets = TOTAL\_NUMBER\_UPPER\_TICKETS;

* public Concert (int m, int d, int y, String a, String v)
  + sets the month, day, year to the provided input parameters – For now, assume the values entered as input parameters are correct
  + sets the artist name and venue to the provided parameters
  + sets all the other instance variables the same way it was done on the default constructor

**Accessor Methods**

An *accessor* method is used to ‘read’ class fields/instance variables. The names for these methods, which simply return the current value of a field, often begin with the prefix ‘get’.

* public String getArtist () - returns the artist name.
* public String getVenue () - returns the venue of the concert
* public double getTicketPrice (char ticketType) - returns the price of the ticket based on the ticket type.

The valid values for ticket type are:

* + 'U', for tickets in the upper section.
  + 'L', for tickets in the lower section
  + 'F', for tickets in the floor section

Example: If ticketType is 'U', this method should return PRICE\_UPPER\_TICKET, etc.

* public int getAvailableUpperTickets () - returns the number of available tickets in the upper sections
* public int getAvailableLowerTickets () - returns the number of available tickets in the lower sections
* public int getAvailableFloorTickets () - returns the number of available tickets in the floor sections
* public double getTotalSales () - returns the concert total sales
* public int getMonth() – returns the month of the concert
* public int getDay() – returns the day of the concert
* public int getYear() – returns the year of the concert

**Mutator Methods**

A mutator method performs tasks that may modify class fields. Methods that simply set a field with the parameter value often begin with the prefix ‘set’.

* public void setArtist (String n) – sets the artist name to the value passed as input parameter
* public void setVenue (String n) – sets the venue to the value passed as input parameter
* public void setDate(int m, int d, int year) – sets the month, day and year to the values of the input parameters. For now, assume the input parameters are correct.

# Phase 2 (30 pts)

**Private Helper Methods**

Private helper methods help public methods carry out tasks.

* private void parseDate (String date) – This method parses the string passed as input parameter into individual pieces and converts them to integers. The date is contained in a string with values separated by slashes. The month and day can contain one or 2 digits. For now, assume the value entered as input parameter is correct.

Samples of data:

"9/8/2019"

"09/8/2019"

"9/08/2019"

"09/08/2019"

Code example to parse the month:

int firstSlash = date.indexOf ("/");

month = Integer.parseInt (date.substring (0, firstSlash));

**Write a third constructor.**

* public Concert (String date, String a, String v)
  + sets the artist name and venue to the provided parameters.
  + invokes the private method that parses the concert date entered as String into 3 int variables associated with the month, day and year of the concert

Example:

parseDate (date);

* + sets all the other instance variables the same way it was done in the default constructor

**Write additional methods.**

* public void buyTickets (char ticketType, int numTickets, double pmt)
  + For now, assume numTickets and pmt contain positive int values
    - If there are available number of tickets (numTickets) in the specific section/ticket type that you want to buy.
      * calculate the amount due based on the ticket type and number of tickets
      * If the payment, entered as input parameter, is large enough to pay for the tickets: 1) add the amount due to total sales, 2) decrease the number of tickets available for the specific section and 3) display the results of the transaction.
      * If the payment, entered as input parameter, is not enough to pay for the tickets: print a message indicating the problem with the amount. The client cannot buy the tickets.
    - If there are NOT available number of tickets (numTickets) in the specific section/ticket that you want to buy, print a message indicating that the client cannot buy the tickets.

The valid values for ticket type are:

* + 'U', for tickets in the upper section.
  + 'L', for tickets in the lower section
  + 'F', for tickets in the floor section
* public void printReport() – prints a daily report of the ticket sales using System.out.println statements. Your report should be similar to the following example. Pay close attention to text alignment and currency format. Use a NumberFormat object to correctly display the monetary value (described below).

Output Examples:

1. printReport method:

Concert Report

======================

Artist: Keith Urban & Kelsea Ballerini

Venue: Van Andel Arena

Date: October 19, 2019

Tickets sold:

=============

Upper: 15 $448.50

Lower: 17 $1,683.00

Floor: 28 $5,040.00

==============

Total Sales: $7,171.50

1. buyTickets method: successful transactions
   * + Transaction: Number tickets Floor Section: 10, total: $1,800.00
     + Transaction: Number tickets Lower Section: 1, total: $99.00
     + Transaction: Number tickets Upper Section: 1, total: $29.90
2. buyTickets method – unsuccessful transactions
   * + Error - invalid (negative) number of tickets or payment
     + Error – payment is not enough to buy the tickets
     + Tickets not available in upper section

**Formatting Numbers as currency**

# NumberFormat currency = NumberFormat.getCurrencyInstance(Locale.US);

System.out.println("Cost: " + currency.format(2.3323));

will display:

Cost: $2.33

# Phase 3 (30 pts)

**Add the following private Helper Methods**

* private boolean isLeapYear(int y) – return true if the year passed as parameter is a leap year. Otherwise return false. Every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but the centurial years (years divisible by 100) are leap years if they are exactly divisible by 400. For example, the years 1700, 1800, and 1900 are not leap years, but the years 1600 and 2000 are.
* private boolean isDateValid (int m, int d, int y) – return true if day, month and year entered as input parameters are valid, return false otherwise. To be able to write the logic for this method you need to know the number of days for each month of the year, whether a year is a leap year (leap years have 29 days for the month of February, not leap years have 28 days for the month of February). Remember to invoke the isLeapYear method when appropriate. Valid months are between 1 and 12 (inclusive) and consider a positive number as a valid year.

Examples:

* + Invalid dates:
    - 2/29/2019 – 2019 is not a leap year, the number of days in February is 28
    - 13/17/2019 – 13 is not a valid month
    - 11/31/2019 – the month of November only has 30 days
    - 6/23/-2019 – the year cannot be negative
  + Valid dates:
    - 2/29/2020 – 2020 is a leap year, February will have 29 days
    - 11/30/2019 – nothing wrong, November has 30 days
    - 6/22/1992 – nothing wrong here.

**Preventing User Error**

A good programming practice is to avoid, or at least to minimize, the effect of user errors.

* Modify the following method to ignore negative values. If the input parameters associated with the number of tickets or payment are negative, print an error message and do not allow the client to buy the tickets.
  + buyTickets
* Modify each of the following methods to ignore invalid dates. Remember to invoke the isDateValid method when needed. The idea is to use the private methods to avoid repeating code. If a date is invalid, print an error message and do nothing to the concert date.
  + Constructors
  + parseDate
  + setDate

**Advance Features**

* public String formatDate(int format)- The format parameter indicates the type of formatted date to be returned. (10 pts)

format = 1 - return a formatted date string (for example "7/10/2019"). Could be one or two digits for the month and day

format = 2 - return a formatted date string always displaying two digits for the month, day and 4 digits for the year (for example "07/10/2019"). You are not allowed to use any decision statements like if or switch. Hint: DecimalFormat class.

format = 3 - return a formatted date with the 3-character month description. So, the date 7/10/2014 would return the String "Sep 10, 2019". You are not allowed to use any decision statements like if or switch. You can do this by creating a single string with all of the 3-character month abbreviations and using the substring method. The day could be one or two digits.

Example: "JanFebMarAprMay…Dec"

format = 4 – return a formatted date with the month description. Use a switch statement to assign the correct month name (for example September 10, 2019). The day could be one or two digits.

* Modify the printReport method to invoke the formatDate method to print the date of the concert with the full month description.
* public void simulateCompanyBuyingTickets(int numberTickets) – Use a loop to simulate a company buying a total number of individual tickets (numberTickets) in different sections. **You must invoke the buyTickets method.**

In the simulateCompanyBuyingTickets method generate the three input parameters needed to invoke the buyTickets method as follows:

* + char ticketType. Use the Random class to generate an integer random number between 1-3 (inclusive), based on this number assign a value to the ticketType.

Example: if the random number is 1, ticketType is 'U', etc.

* + int numTickets. Set this input parameter to 1.
  + double pmt. Value of one ticket for the section/ticket type generated at random.

Example:

If buying a ticket in the upper section, set pmt to PRICE\_UPPER\_TICKET, etc.

# Coding Style (10 pts)

Good programming practice includes writing elegant source code for the human reader. Follow the GVSU [Java Style Guide](http://www.cis.gvsu.edu/java-coding-style-guide/).

# Software Testing (10 pts)

Software developers must plan from the beginning that their solution is correct. BlueJ allows you to instantiate objects and invoke individual methods. You can carefully check each method and compare actual results with expected results. However, this gets tedious and cannot be automated.

**Testing Your Class using the main( ) method**

Another approach is to write a main method that calls all the other methods in a carefully designed sequence. See section 5.12 in the zyBook.

For this project, write a main method in a new class called ConcertTest that instantiates at least two concerts and invokes several methods with a variety of parameter. Provide multiple print statements and if statements to test the methods along with error messages as needed.

A brief example is provided below. Your test method should be much longer. A thorough test suite will have more test cases (if statements).

public static void main(String args[]){

System.out.println("Testing starts");

int errors = 0;

// instantiating a concert

Concert c1 = new Concert(10,19,2019,

"Keith Urban & Kelsea Ballerini",

"Van Andel Arena");

// buy 10 tickets in the floor section

c1.buyTickets ('F', 10, 1800.0);

if(c1.getTotalSales() != 1800.0){

errors++;

System.out.println("ERROR: Total Sales should be 1800.0");

}

// the floor tickets available should be 390

if(c1.getAvailableFloorTickets() != 390){

errors++;

System.out.println("ERROR: available floor tickets should be 390");

}

// buy 1 ticket in the floor section

// testing errors in input parameters

c1.buyTickets ('F', -1, 180.0);

if(c1.getTotalSales() != 1800.0){

errors++;

System.out.println("ERROR: Total Sales should be 1800.0");

}

// invoking the method to simulate a company buying 50

// individual tickets in random sections

c1.simulateCompanyBuyingTickets(50);

// invoking the report method

c1.printReport();

System.out.println("Testing Complete. Number of errors: " + errors);

}

The printReport method from the code above should produce a similar output. The sum of the number of tickets sold in the different sections will be 60. The dollar amounts will not be the same because of the randomness of the simulateCompanyBuyingTickets method.

Concert Report

======================

Artist: Keith Urban & Kelsea Ballerini

Venue: Van Andel Arena

Date: October 19, 2019

Tickets sold:

=============

Upper: 15 $448.50

Lower: 17 $1,683.00

Floor: 28 $5,040.00

==============

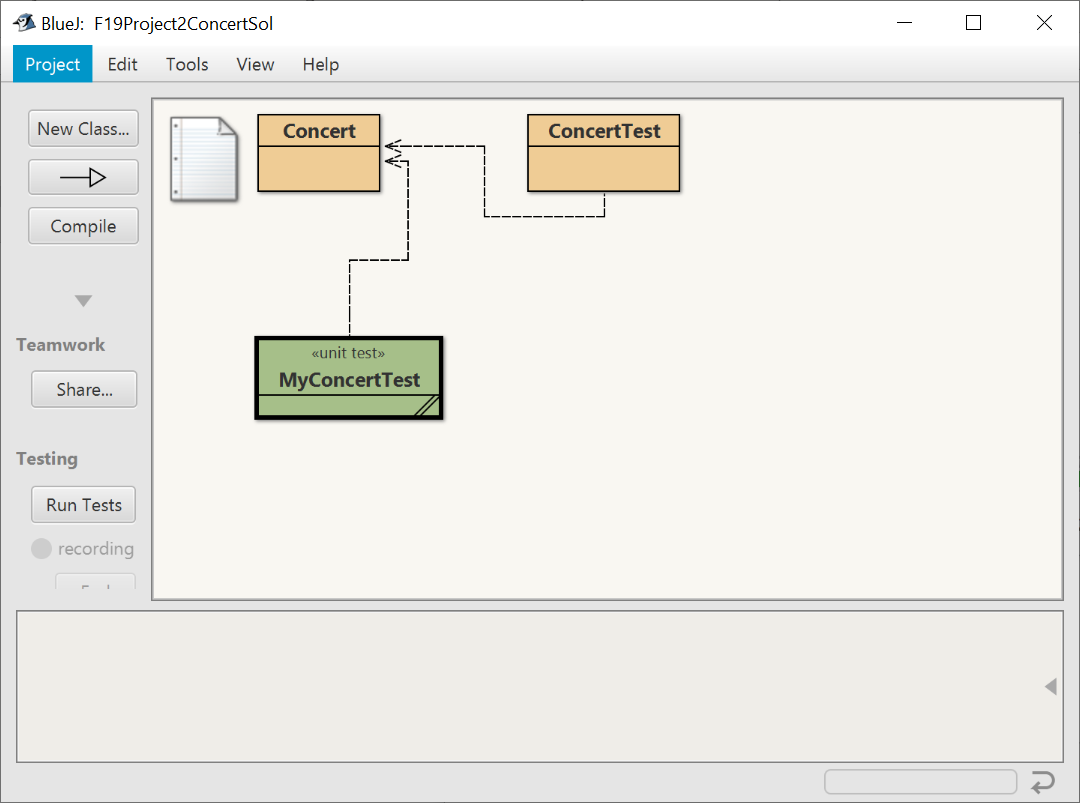
Total Sales: $7,171.50

# JUnit Testing

JUnit is a Java library that provides a framework to automate software testing. A JUnit test file MyConcertTest.java has been provided by your instructor. Use the following instructions to run JUnit Testing

1. **Exact spelling is required** for the name of the Concert class and the headers of all the methods specified above.
2. Complete all the project requirements.
3. Download MyConcertTest.java to the same folder as your Concert.java

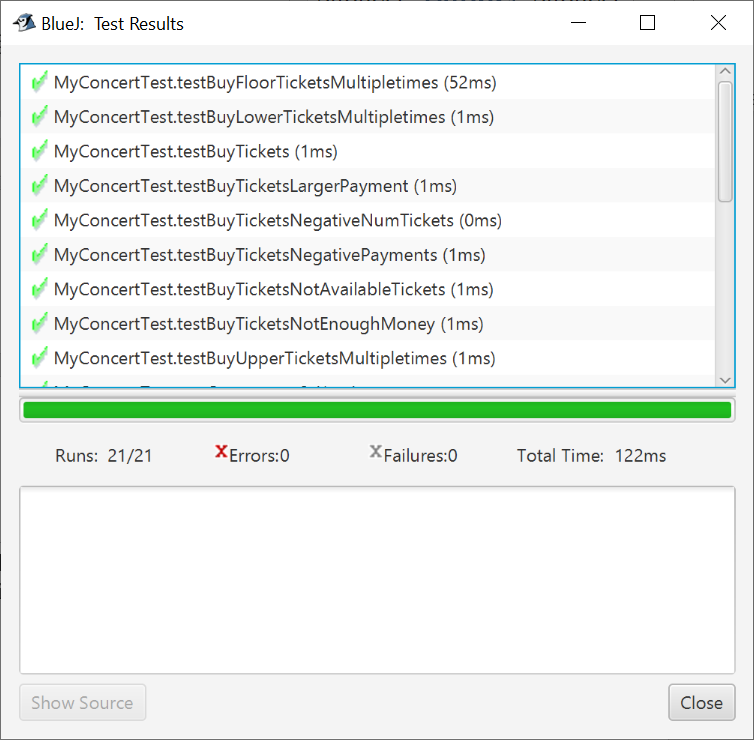
Restart BlueJ (if MyConcertTest does not show up on your BlueJ window). BlueJ should recognize MyConcertTest as a “<<unit test>>” file.



1. On the BlueJ window, click the “Run Tests” button.

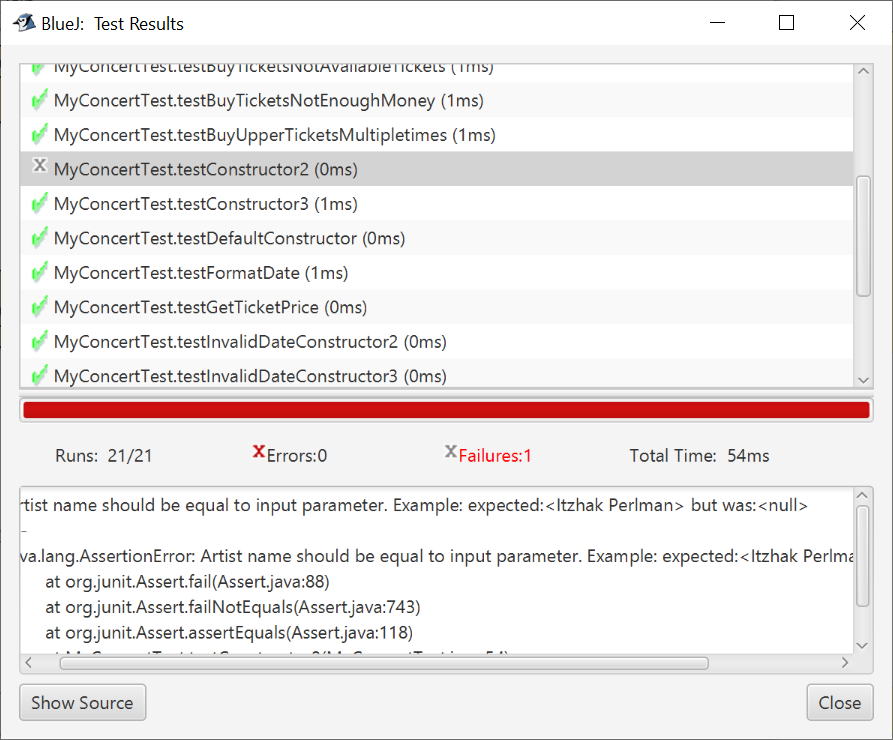
Example: JUnit test passes all the test cases

If your class passes all the test cases, the test results window will show a green horizontal bar and zero failures.



Example: JUnit test does not pass all the test cases:

If your class does NOT pass all the test cases, you will see a red horizontal bar and the number of failures. Click on a test result to show a longer description of the failure.



# Grading Criteria

* Project requirements as specified above (100 pts).

# Late Policy

Projects are due at the START of the lab period. However, you are encouraged to complete a project even if you must turn it in late.

* The first 24 hours (-20 pts)
* Each subsequent weekday is an additional -10 pts
* Weekends and university holidays are free days.

# Turn In

A professional document **is stapled** with an attractive cover page. Do not expect the lab to have a working stapler!

* Cover page - Provide a cover page that includes your name, a title, and an appropriate picture or clip art for the project
* Signed Pledge – The cover page must include the following signed pledge: "I pledge that this work is entirely mine, and mine alone (except for any code provided by my instructor). " In addition, provide names of any people you helped or received help from. Under no circumstances do you exchange code electronically. You are responsible for understanding and adhering to the [School of CIS Guidelines for Academic Honesty](http://www.cis.gvsu.edu/academic-honesty/).
* Time Card – The cover page must also include a brief statement of how much time you spent on the project. For example, “I spent 7 hours on this project from September 22-27 reading the book, designing a solution, writing code, fixing errors and putting together the printed document.”
* Sample Output – a printout of the BlueJ Terminal window after running the main method that shows a variety of the printed messages. You can copy and paste into the Word document that contains your cover page.
* Source code – DO NOT PRINT - upload to Blackboard the source code of the two classes.

Concert.java

ConcertTest.java

* Demo – be prepared to demo your project on a lab computer or your laptop. I will ask you to run the JUnitTest class – MyConcertTest.java.