

# PCS4 EXAM – JUNE 2018

**DATE: 22 JUNE 2018**

**TIME: 12.45 – 15.15 H (150 MINUTES).**

## ADMITTED RESOURCES:

- You are allowed to use everything on paper (books, notes, etc.) and on your laptop, but only what you bring in: you are not allowed to borrow something from someone else.
- During the exam, it is not allowed to use the network. You should make the exam yourself: so, no communication with MSDN or google for help and no communication with other students, like using facebook, e-mail, skype, Dropbox, gsm or whatever.

**GRADING: SEE ASSIGNMENTS.**

## THE APPLICATION

With this application the store owner can manage and keep track of available songs for sale. New songs can be added to the list. The price of a song can be changed later.

The application also offers a way to notify fans for newly added songs. The application also offers a way to notify for song price changes.

There is also a promo board where promoted songs are displayed.

The application has three forms, the main form for the store owner, a form for Beatles fan and a form for promoted songs.

## MAIN FORM

The screenshot shows a window titled "The Nuke Juke" with a purple background. The interface is divided into several sections:

- Sort Section (Top Left):** Two orange buttons labeled "Sort by Year" and "Sort by Artist, Price and Name".
- Add Song Section (Green):** A form with fields for "Name" (containing "Blackbird"), "Artist" (containing "The Beatles"), "Year" (containing "1968"), and "Price" (containing "7"). Below the fields is a blue button labeled "Add Song to Store".
- Find Cheapest Beatles Song Section (Yellow):** A button labeled "Find Cheapest Beatles Song".
- Nr of plays per artist Section (Light Blue):** A form with a label "Nr of plays per artist", a text field "Artist / Band" containing "Rolling Stones", and a blue button labeled "Nr of credits per Artist".
- Song List Section (Purple):** A list of songs with their artists, years, and prices. The list includes:
  - Another Brick in the Wall, Pink Floyd, 1979 | €2.5
  - Hey Jude, The Beatles, 1968 | €3
  - Candle in the Wind, Elton John, 1997 | €5.5
  - Umbrella, Rihanna & Jay-Z, 2007 | €5
  - Heart of Glass, Blondie, 1979 | €2.8
  - Paint It, Black, Rolling Stones, 1966 | €3.3
  - Here Comes the Sun, The Beatles, 1969 | €3.7
  - Billie Jean, Michael Jackson, 1983 | €4.2 (highlighted in blue)
  - Shape of My Heart, Sting, 1993 | €5
  - Don't Let Me Down, The Beatles, 1988 | €3.9
  - Lose Yourself, Eminem, 2003 | €3
  - Nothing Else Matters, Metallica, 1991 | €5.25
  - Redemption Song, Bob Marley, 1980 | €2.2
  - Yesterday, The Beatles, 1965 | €5
- Change Price Section (Bottom Right):** A form with a label "Change Price", a text field containing "1.8", a blue button labeled "New Price", and a blue button labeled "Play".
- Test data added Section (Bottom):** A red bar with the text "Test data added".

Figure 1

In *Figure 1* we can see the main window `JukeboxForm` that displays the application used by the store owner.

In the top left corner, there are two orange buttons for sorting the list. Underneath that in the green section there are controls for adding a new song. A button to find the cheapest song from Beatles. In the bottom left corner there is a section with controls for finding how many credits an artist/band has earned.

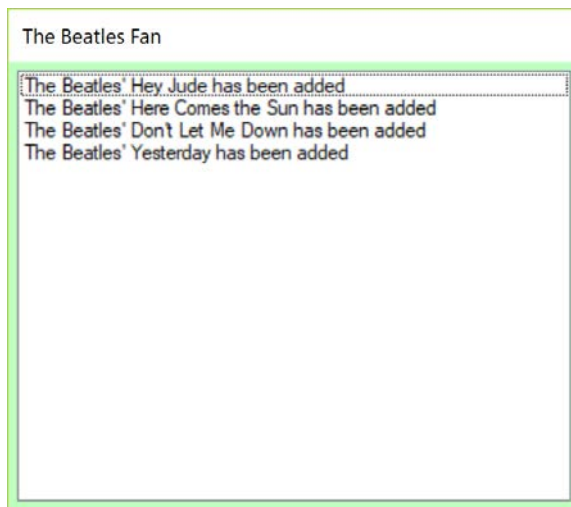
On the right side, we have the purple section that shows the list of available songs for sale, the button for changing the price of a selected item and a button for simulating playing a song.

In the bottom right there is a label that could be used to display information.

The application also has two other forms which are described below.

## THE BEATLES' FAN FORM

---



The Beatles Fan

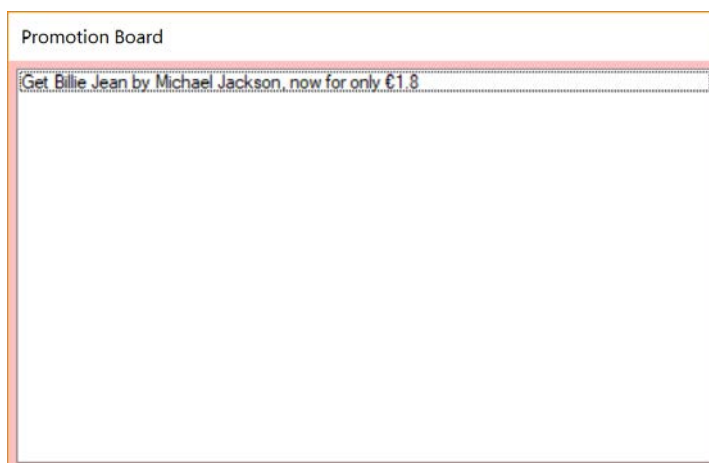
The Beatles' Hey Jude has been added  
The Beatles' Here Comes the Sun has been added  
The Beatles' Don't Let Me Down has been added  
The Beatles' Yesterday has been added

In *Figure 2* an example of the second form is presented. This form lists only information that the client (Beatles fan) is interested about.

Figure 2

## THE PROMO BOARD FORM

---



Promotion Board

Get Billie Jean by Michael Jackson, now for only £1.8

In *Figure 3* an example of the promotion board that is used to promote cheap songs of any artist.

Figure 3

## PRELIMINARY REMARKS

---

- Whenever this exam paper suggests using a certain name for a method, variable or anything else, you are required to use exactly that name.
- The classes `Song`, `JukeboxStore`, `JukeboxForm`, `BeatlesFanForm` and `PromoBoardForm` already contain some code. It will be your task to add more code to these classes, where necessary. You may add whatever you need. It will also be your task to add more classes, if needed.

Worth noting is that `JukeboxForm` has 3 important variables:

```
private JukeboxStore myStore;           // The store with the list of songs
private BeatlesFanForm frmBeatles;      // The form for Beatles fans
private PromoBoardForm frmPromo;        // The form for promoting cheap songs
```

As seen above, the `JukeboxForm` window has several buttons with text "Add Song to Store", "New Price", and "Find Cheapest Beatles Song", that are all partially implemented.

It will be your task to complete the functionality of these buttons and add the functionality to other buttons based on the requirements below.

*In order to help with testing the implementations, you may but you are not required to use the method `private void AddTestData()`. This method adds some songs as shown in the [Figure 1](#).*

#### ASSIGNMENT 1: (10+15PTS).

Consider the orange colored buttons with the text "Sort by Year" and "Sort by Artist, Price and Price".

##### ASSIGNMENT 1A:

Clicking on the button with the text "Sort by Year" should sort the list of the songs of `myStore` from the oldest to the most recent year.

The button click is already implemented by calling `myStore.SortByYear();` method. However, the method `SortByYear` has no code.

Implement the method `public void SortByYear()` in the `myStore` class by using one of the 3 ways studied during PCS4 (so by using `IComparable`, `Comparison` or `IComparer`).

##### ASSIGNMENT 1B:

Clicking on the button with text "Sort by Artist, Price and Name" should sort the list of the songs of `myStore` by "Artist, Price and Name". The songs must be sorted by artist alphabetically. If songs have the same artist, then they must be sorted by price from the most expensive to least expensive (so by price in descending order). If the songs have the same artist and price, then they should be sorted alphabetically by name.

Again, in the `JukeboxForm` this button-click is already calling a method as `myStore.SortByArtistPriceAndName()`, but the method has no body. To make it work give an implementation of `public void SortByArtistPriceAndName()` by using one of the 3 ways, but **not** in the same way as you did in assignment 1A.

## ASSIGNMENT 2: (11+12+12 PTS).

In the main form the green buttons with text "Add Song to Store" and "New Price" are partially implemented as follows:

- When clicking "Add Song to Store", a new Song will be created using input from the corresponding textboxes and then added to the list of Songs in `myStore`.
- Clicking on "New Price" first checks if a song is selected in `listbox 1bSongs`. If no song is selected before pressing the button, then a warning message is shown, otherwise the Price of the selected song is changed to the inputted value.

As mentioned earlier, in some cases fans or potential customers should be notified if a change occurs. Adjust the code in such a way, that the following functionality works correctly. For each part of this assignment, you must give a solution by using events. Implementations without using events will not be rewarded.

### ASSIGNMENT 2A:

If a song by The Beatles is added to the list, fans of The Beatles would like to be notified. A message should be displayed on the Beatles Fan form (`frmBeatles`). The form already has a method for this, which show information about the song and the reason for the notification (in this case the reason is that a song has been added):

```
public void InformAboutBeatlesSong(Song s, string reason)
```

*For example, consider the situation as depicted in Figure 1 and Figure 2:*

When the button with text "Add Song to Store" is clicked, Beatles fans should be informed, since a new song from The Beatles (in this case the song Blackbird) is offered for sale.

### ASSIGNMENT 2B:

Beatles fans also want to be notified if the price of any Beatles song changes, regardless of the value. This should also be displayed on the Beatles Fan form using the same method (now the reason is that the price of the song has changed):

```
public void InformAboutBeatlesSong(Song s, string reason)
```

*For example, consider Figure 2:*

The songs "Blackbird" and "Hey Jude" have had their prices changed.

### ASSIGNMENT 2C:

If the price of any existing song by any artist changes from a price higher than € 2 to a new price of € 2 or less, this should be displayed on the promotion board form (`frmPromo`) using the method:

```
public void PromoteCheapSong(Song s)
```

*For example, consider the situation in Figure 1 and Figure 3:*

When the button with text "New Price" is clicked, the price of the selected song ("Billy Jean" by Michael Jackson) is changed from € 4.2 to € 1.8. As the price of the song is now less than or equal to € 2, it is displayed on the promotion board.

### ASSIGNMENT 3: (18 + 2 PTS).

#### ASSIGNMENT 3A:

In the class `JukeboxStore` we have a method `public double GetCheapestBeatlesSong()`

This method should return price of a cheapest Beatles song, if there is no Beatles song, return 9999. You may assume that no song has price bigger than or equal to 9999.

Give an implementation by using recursion.

A solution without using recursion will not be rewarded.

#### ASSIGNMENT 3B:

Implement the functionality of the button with text "Find Cheapest Beatles Song".

Clicking this button should result in displaying the minimum price of all available songs with artist "The Beatles".

Example:

Assume the list of songs for sale is as displayed in *Figure 1*. Pressing the button with text "Find Cheapest Beatles Song", results in showing **2**, since that's the price of the cheapest Beatles song.

#### ASSIGNMENT 4: (14+4+2 PTS).

Every time a song of an artist is played, that artist will earn a credit. We would like to keep track of that using a linked list. Every time a song is played, it will be added to the linked list. The song might be added to the list more than once if it's played multiple times.

##### ASSIGNMENT 4A:

Add a class `LLOfPlayedSongs` to your solution (*LL stands for linked list*). This class must implement a linked list of songs that are played. The class must have at least these two methods, described below:

The method:

```
public int NrOfCreditsPerArtist(string artistName)
```

should return the total number of credits of a particular artist with name `artistName` based on the number of occurrences in the linked list.

The method

```
public void RegisterPlayed(Song s)
```

should add the song `s` to the linked list.

##### ASSIGNMENT 4B

- Add a variable of type `LLOfPlayedSongs` to your `JukeboxForm` class.
  - Clicking the button with text "Play" should add the song that is selected from the listbox `lbSongs` to the linked list.
- We don't really play the song: that would be too noisy during this exam.

##### ASSIGNMENT 4C

When clicking the button with text "Nr of credits per Artist", the total number of times played for all songs of that artist (specified in `tbArtistPlayed`) will be shown.

Give an implementation for:

```
private void BtnNrOfCreditsPerArtist(object sender, EventArgs e).
```

**END OF EXAM.**