Exam PCS1 October 31, 2018

Time: 150 minutes

Important notes:

- 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, mobile phone or whatever.
- If the code you submit does not compile, you will get 0 points! Make sure your app can be started, even if not all functionality has been implemented.

+-

Let's go bowling!

Introduction.

Your assignment for this exam is to create an app that can be used to keep track of the scores in a bowling match. Two players can play a match, or one player can play against a computer opponent.

We use a simplified version of the rules for 10-pin bowling: every round, each player throws the ball once, the score for that round is simply the amount of pins knocked over. A player's total score is the sum of their scores for each round.

The user interface for this app has already been made (see Screenshot 1), you can find it in the startup project.

You will need to add the class Player to make the app work, see Diagram 1. More specific instructions will be provided in the assignments. In case you need to add more members, feel free to add whatever you need.

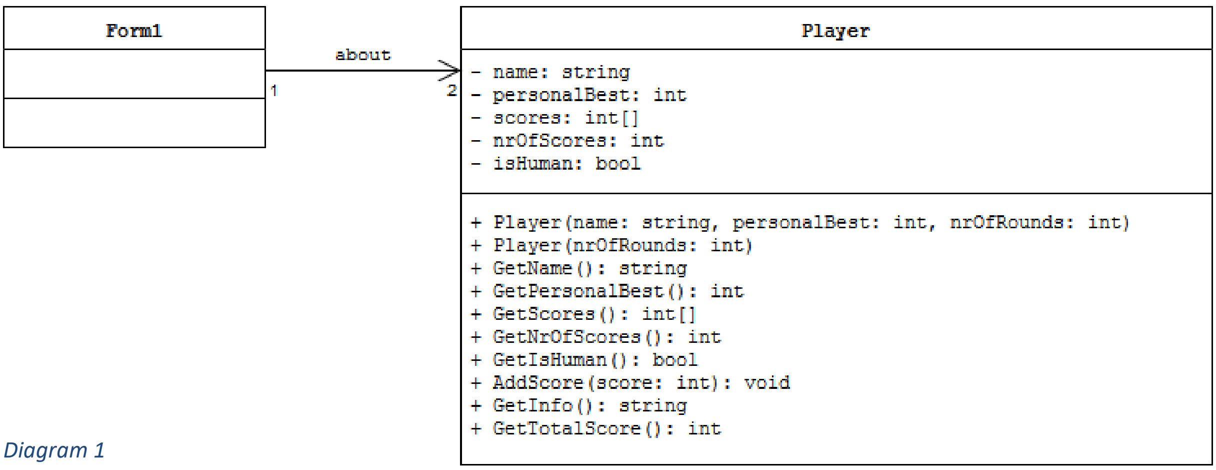
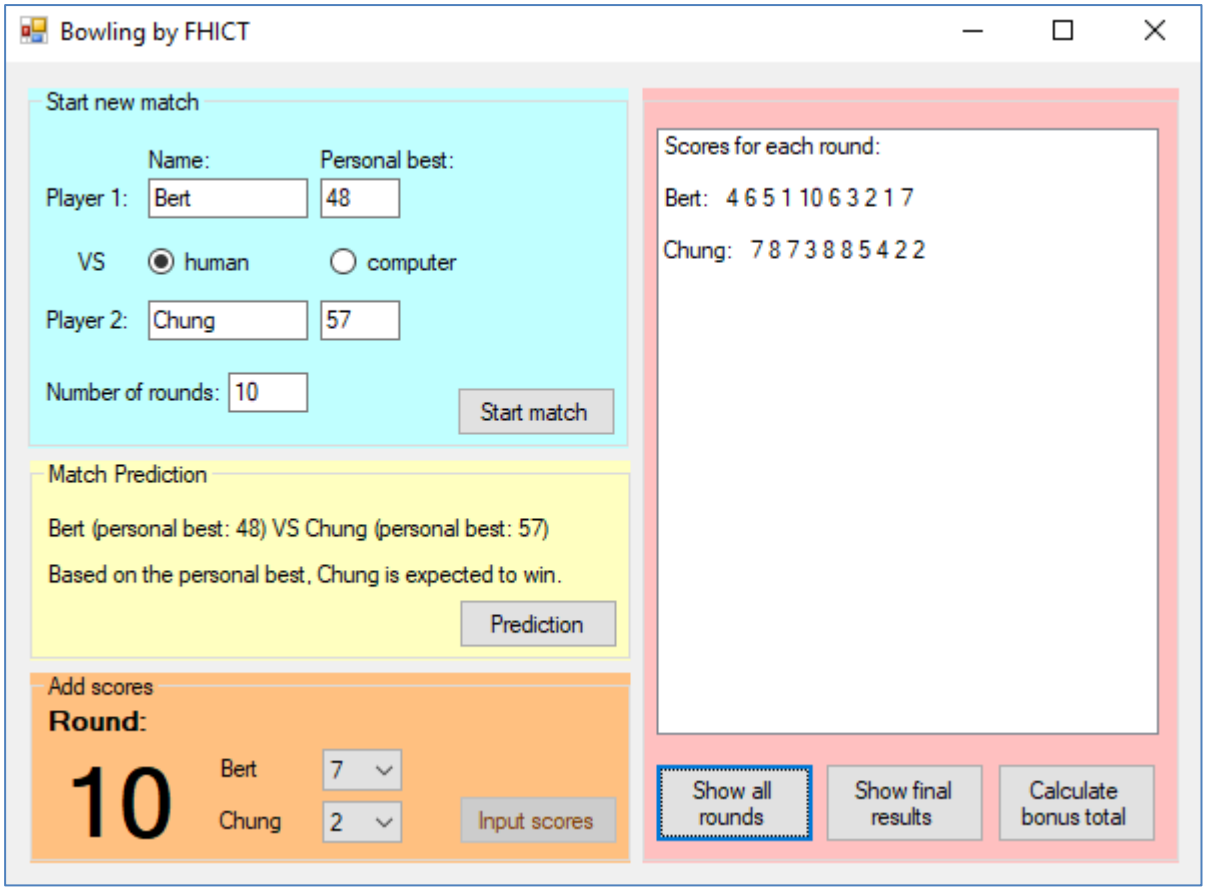


Diagram 1

The user interface (Form1) looks like this:



Screenshot 1

On the left, there are three groupboxes, that each contain the controls for a specific function:

- The blue groupbox marked “Start new match” is used to set the names and personal best (highest score they ever achieved in a match) of the players, set the number of rounds in a match and start a new match. A player can also play a match against a computer opponent.
- The yellow groupbox marked “Match prediction” is used to show information about the players and predict who will win based on their personal best.
- The orange groupbox marked “Add scores” is for inputting the score after every round. For human players, a score from 0 to 10 can be input, for a computer player a random score in that range will be added.
- On the right side of the form there is a listbox for displaying information about the match. There are also some buttons that will be discussed in the assignments.

Now it’s time to open the startup project, you will need to implement the Player class (assignment 1) and the Form1 class (assignment 2).

Assignment 1: The "Player" class (6 + 6 + 6 + 6 + 8 + 8 = 40 points)

Add a Player-class to the project, see Diagram 1 on page 1 for the specification and read below for additional information.

Assignment 1a:

First of all, add some fields to the class as specified in Diagram 1. About those fields:

- **name** is the name of the player
- **personalBest** is the highest score the player has ever achieved
- **scores** is an array to register the player's score for each round
- **nrOfScores** represents how many scores have been entered for this player
- **isHuman** indicates if the player is a human player or a computer

Assignment 1b:

Implement two constructors for this class

- the first constructor is used to create human players, it takes three parameters: the inputted player name and personal best and the number of rounds in a match
- the second constructor is used to create a computer player, this only takes the number of rounds in the match as a parameter. The name should be set to "Computer" and the personal best to the number of rounds multiplied by 5 (an average score).

Assignment 1c:

Implement public methods *GetName*, *GetPersonalBest*, *GetScores*, *GetNrOfScores* and *GetIsHuman* that return the values of the corresponding fields.

Assignment 1d:

Implement the *AddScore* method as specified in Diagram 1. If possible, this method should add a score to the array.

Assignment 1e:

Add a method *GetInfo* to the *Player* class. For a human player, this method should return a string containing the name and personal best, so something like "Bert (personal best: 50)". For a computer player, this string only contains the name (so no personal best).

Assignment 1f:

Add the method *GetTotalScore* (see Diagram 1). This method should return the total score for the player up until that moment. No bonus effects are taken into account yet, so this is just the sum of the scores for each round.

Assignment 2: The "Form1" class (10 + 10 + 10 + 10 + 10 + 10 = 60 points)

Assignment 2a:
First of all, at the top of the window in Screenshot 1, you see in the title-bar the text `"Bowling by FHICT"`, change this text to include your own name.

The blue groupbox is used to start a new match.

- If the button with text "*Start match*" is clicked, two players are created with the details supplied in the other controls in the blue groupbox. Player 1 is always a human player, the radiobutton determines if player 2 is a human or a computer opponent. Don't forget to declare the necessary variables for your Player objects. Users can choose the number of rounds in the match by supplying a number in the bottom textbox.
- Upon clicking the button, the labels in the orange groupbox marked "*Add score*" should change from "*Name 1*" and "*Name 2*" to the actual names of the players.
- In case player 2 is a computer opponent, also disable the combobox for player 2's score in the orange groupbox.

Implement the method: `private void btnStartMatch_Click(object sender, EventArgs e)`

Assignment 2b:
The yellow groupbox is used to make a prediction about who will win the match, like in the example in Screenshot 1. There are two labels in this groupbox, the top label is for showing some information about the players, the bottom label is used to display a prediction about who will win. The player with the highest personal best should be the predicted winner, if both players have the same personal best, the bottom label should display that the match could go either way.

When the button marked *"Prediction"* is clicked, the required information must be displayed in the labels.

Implement the method: `private void btnPredict_Click(object sender, EventArgs e)`

Assignment 2c:
In the orange groupbox, there are two comboboxes that can be used to enter the score of each player. There is also a button with the text "*Input scores*". Clicking this button should:

- Register the scores of this round for both players in the arrays of both players. The score for player 1 is alsways indicated in the combobox *cmbScorePlayer1*. If player 2 is a computer opponent, instead of reading the value from the combobox, a random number at least 0 and at most 10 should be added instead. In this case, the combobox for player 2 is ignored.
- Show the next round number in the label *lblRoundNr.*
- Disable the button after the final round.

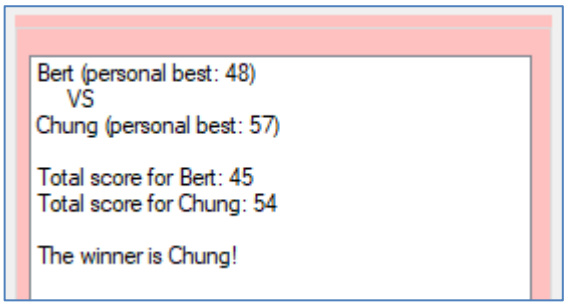
Implement the method: `private void btnInputScores_Click(object sender, EventArgs e)`

Assignment 2d:
Clicking the button with text "*Show all rounds*" should result in showing the names of the players and then their scores for all rounds, all together on a single line, like in the example in Screenshot 1.

Implement the method: `private void btnShowRounds_Click(object sender, EventArgs e)`

Assignment 2e:
Clicking the button with text "*Show final results*" should display an overview of the match like in Screenshot 2. If both players have the same score, the final line should be "*It's a tie!*".

If this button is clicked before the end of the match, it will display who would be the winner, if this was the end of the match.

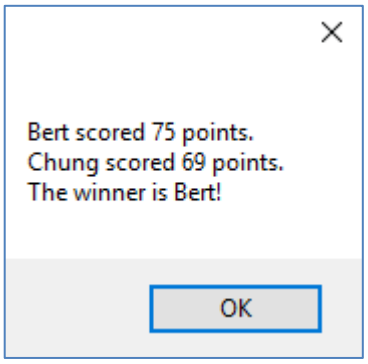


Screenshot 2

Implement the method: `private void btnShowResults_Click(object sender, EventArgs e)`

Assignment 2f:
In real bowling, when a player throws a strike (10 points), his score for the next round counts double. (example: player 1 scores 10 points in round 3, 10 points in round 4, and then 8 points in round 5. Now his scores for round 4 and 5 count for 20 points and 16 points respectively). To keep things simple, throwing a strike in the final round does not give any extra points.

Clicking the button with text "*Calculate bonus score*" should result in calculating the total scores in accordance with this rule and displaying them on screen like the example in Screenshot 3.



Screenshot 3

Make this work by implementing: `private void btnBonus_Click(object sender, EventArgs e)`

END of pcs1-exam.