Dostihy a sázky

Ročníková práce

Programování – 4.E

Autor: Ondřej Záhorský

Škola: Gymnázium, Praha 6, Arabská 14

Kraj: Praha

Konzultant: Mgr. Jan Lána

Třídní učitel: Mgr. Barbora Novosadová

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 27. 3. 2021 …….

Anotace

Cílem této ročníkové práce bylo naprogramovat počítačovou verzi hry „Dostihy a Sázky“. Jedná se o deskovou hru pro dva až čtyři hráče. Na začátku hry se objeví hráčské figurky na startovním poli a každý hráč má k dispozici 20 000 virtuálních peněz. Hráči postupně hází kostkou a pohybují se svými figurkami. Jakmile hráč vstoupí na pole, kde se nachází kůň nebo trenér a nikdo dané pole nevlastní může si koně nebo trenéra koupit. Když hráč vlastní všechny koně ve stáji může si začít nakupovat dostihy. Čím více má hráč koupených dostihů na určitém koni, tím větší má kůň cenu. Hráč samozřejmě může trenéry, koně i dostihy prodat, a to kdykoliv je na tahu, nehledě na to, kde se nachází jeho figurka. Pokud se hráč zastaví na poli, které má koupené protihráč, musí mu zaplatit určitou cenu. Jestliže hráč nemá peníze, zkrachuje a hra pro něj končí. Zároveň pokud už zbývá jenom jeden hráč, hra skončí a vypíše pořadí hráčů.

The goal of this seminar work was to program game „Racing and Betting“. Racing and Betting is a board game fot two to four players. At the start of the game player’s figure appear on the starting field and each player has 20,000 virtual money at their disposal. Players gradually roll the dice and move their figure. Once a player enters a field where a horse or coach is located and no one owns that field, he can buy a horse or coach. When a player owns all the horses in the stable, he can start buying races. The more a player has purchased races on a particular horse, the greater the price of the horse. Of course, a player can sell coaches, horses and races at any time, regardless of where his figure is. If a player stops on the opponent's field, he must pay him a certain price. If a player has no money, he goes bankrupt and the game ends for him. At the same time, if there is only one player left, the game ends and ranks of players will appear on the screen.

Obsah

1. Úvod

a. Očekávání

Na začátku práce jsem se rozhodoval, v jakém programovacím jazyce budu hru vytvářet. Nakonec jsem se rozhodl pro C#, který mi pro můj účel přišel ideální. Také jsem chtěl vytvořit grafiku, která se bude odvíjet od originální verze. Dále jsem si uvědomoval, že práce bude časové náročná a budu si muset práci dobře časově naplánovat.

b. Problémy k řešení

U hry jako jsou Dostihy a sázky je spousta možností co může hráč dělat a také co musí vyhodnotit hra sama o sobě. V aplikaci jsem řešil následující problémy a postupoval následovně:

1) Vytvoření, vykreslení herního plánu, umístění karet na plochu
2) Nastavení informací, které potřebujeme vědět o kartách
3) Vytvoření kostky
4) Vytvoření hráčů
5) Vykreslování karet koní a trenérů v okně
6) Problematika pohybu hráčů po hrací desce
7) Problematika střídání hráčů
8) Problematika nákupu a prodeje karet
9) Problematika nákupu dostihů
10) Problematika konce hry

2. Programovaní hry

Samotný kód hry je složen ze tří projektů *BettingOnHorse*s, *BettingOnHorsesLogic* a *BettingOnHorsesModel*, které se navzájem propojují. Aplikaci jsem programoval v Microsoft Visual Studiu a v tomto prostředí se mi až na pár problému s nastavením programu programovalo velmi dobře.

a. BettingOnHorsesModel

Tento projekt obsahuje celkem deset tříd, přičemž všechny z nich obsahují informace, které pak potřebujeme dále ve hře. Přitom samotné hodnoty do těchto tříd nastavujeme v jiných třídách a metodách. Třída *Card.cs* je třída představující kartu a určujeme v ní hodnoty jako je jméno karty, hodnota karty, kde se karta nachází na desce a dále obrázek jaký má karta mít. Třída *GameStatus.cs* je výčtový typ představující status hry. Jestli je hra aktivní, nebo neaktivní a zdali už skončila. Třída *Player.cs* představuje hráče a definujeme v ní jméno a barvu hráče, kde se hráč nachází, jeho zůstatek, zároveň zjišťujeme, jestli je hráč stále ve hře a také se ptáme, zda hráč házel kostkou a jaké je jeho pořadí ve hře. Třída *Stable.cs* představuje stáj a definujeme v ní barvu stáje a počítáme koně ve stáji.

Další třídy jsou *HorseCard.cs, ParkingCard.cs, StartCard.cs a TrainerCard.cs,* všechny tyto třídy dědí z třídy *Card.cs,* ve třídách *ParkingCard.cs, StartCard.cs* se vlastně nic neděje, protože na těchto kartách není nic k nastavování, u třídy *TrainerCard.cs*, která představuje kartu trenéra se přiřadí ke kartě obrázek a cena karty, která je u všech trenérů stejná a to 4000. A nejobsáhlejší třída z výše zmíněných *HorseCard.cs* je třída představující kartu koně. V této třídě nastavujeme zařazení koně do stáje, jakou má cenu dostih, kolik je na koni nakoupených dostihů, cenu, kterou musí zaplatit protihráč, pokud na danou kartu vstoupí a v neposlední řadě přiřadíme obrázek a nastavíme aktuální počet dostihů na nula.

Jedna z posledních tříd v tomto projektu je *Board.cs* a to je třída představující hrací plochu. V této třídě řešíme problematiku kostky. A druhou metodu, kterou zde máme je metoda *GetLocation*, která vrátí souřadnice pro kartu nebo hráče podle indexu, který je předán v parametru.

A poslední třída v tomto projektu je třída *Game.cs*, což je třída představující hru. Dává nám seznam hráčů, kteří jsou ve hře a také říká, který hráč je na tahu.

b. *BettingOnHorsesLogic*

Tento projekt obsahuje pouze dvě třídy, stará se o připravení hry, následné operace, které se ve hře odehrávají a kontroluje, zdali hra skončila nebo ne. První třída je *TurnResult.cs*, která představuje průběh hry, kontroluje, zdali hra skončila a jestli hráč prošel startem. Druhou třídou je *GameService.cs* a ta je podstatně obsáhlejší než třídy předchozí.

Metoda *CreateGame* připravuje hru pro počet hráčů, který dostane v parametru. Nastaví hráčům a kartám hodnoty a barvy, zavolá funkci *GetLocation*, která jim přidělí pozici na hrací ploše. Připraví stáje a přiřadí je kartám koní. Dále také určí, jaký hráč je na tahu. Třída také obsahuje metodu *MovePlayer*, která přemístí hráče podle jeho hodu kostkou. Vrátí instanci *TurnResult* a tím se dozvíme, jestli hráč prošel startem. V případě že hráč dojde na konec hrací plochy musíme ho vrátit na začátek.

```
public TurnResult MovePlayer(Game game, Player player, int rolledNumber)
{
    var result = new TurnResult() { GameOver = false, StartPassing = false };

    //Pokud dojdeme na konec hrací plochy, musíme se vrátit na začátek
    if (rolledNumber+player.BoardIndex > game.Board.Count-1)
    {
        player.BoardIndex = (player.BoardIndex + rolledNumber) - game.Board.Count;
        player.BoardLocation = Board.GetLocation(player.BoardIndex);
        result.StartPassing = true;
    }
    else
    {
        player.BoardIndex += rolledNumber;
        player.BoardLocation = Board.GetLocation(player.BoardIndex);
    }
    return result;
}
```

Obrázek 1- Přemístění hráče

Další metoda v této třídě je metoda *NextPlayer*, která předává tah dalšímu hráči. V této metodě se řeší že pokud dojdeme na konec listu s hráči musíme se vrátit na začátek a také se ptáme, zda je hráč stále ve hře a pokud ne zavoláme metodu znovu a tah zde předáme dalšímu hráči.

```
public void NextPlayer(Game game)
{
    //Pokud dojdeme na konec listu s hráči, musíme se vrátit na začátek
    if(game.PlayerOnTurn.GameIndex == game.Players.Count - 1 )
    {
        game.PlayerOnTurn = game.Players.Where(x => x.GameIndex == 0).FirstOrDefault();
        game.PlayerOnTurn.DiceRolled = false;
    }
    else
    {
        game.PlayerOnTurn = game.Players.Where(x => x.GameIndex == (game.PlayerOnTurn.GameIndex + 1)).FirstOrDefault();
        game.PlayerOnTurn.DiceRolled = false;
    }

    //Zjistíme, zda-li je hráč aktivní. Pokud není, zavoláme znovu metodu NextPlayer
    if (!game.PlayerOnTurn.IsActive)
    {
        NextPlayer(game);
    }
}
```

Obrázek 2- Předání tahu

Dále je zde také metoda *GetMyCards*, která vrátí list s kartami požadovaného typu, které vlastní hráč převzaný z parametru. Toto používáme například, když chceme zjistit, jestli má hráč kompletní stáj. Což je další metoda v této třídě a funguje na principu, že dostane kartu koně a zjistí, zdali hráč vlastní všechny koně ze stáje. A nakonec metoda *IsGameOver* což je metoda pro zjištění, zdali hra pořád běží. V této metodě zjišťujeme, zda jsou ve hře aktivní alespoň dva hráči a pokud ne hra skončí.

c. BettingOnHorses

V tomto posledním projektu řešíme celou grafiku hrací plochy. Ať už zobrazení hrací desky, zobrazení karet koní, trenérů anebo výsledků hry, vše je řešeno vyskakovacím oknem. Veškerou grafiku řešíme pomocí *Gridů* a *StackPanelů*. U *gridů* nastavujeme pozici jednotlivých prvků na ploše. A do *StackPanelů* dáváme prvky, které chceme vypsat. Zároveň máme *v Resourses* v tomto projektu uloženy veškeré obrázky, které používáme.

U všech karet máme v základu nastaveny v grafice obecné parametry a specifické hodnoty nastavujeme zvlášť u jednotlivých karet. Při vypsání specifických hodnot u karet se ptáme, zdali je karta vlastněna některým z hráčů a pokud ano, tak vypíšeme jeho jméno. Nastavení karet trenérů a koní je hodně podobné s tím rozdílem, že u karet koní řešíme navíc přítomnost dostihů. U obou karet je nastavení takové, že pokud na kartě stojí hráč, který je na tahu a karta ještě nemá vlastníka, zobrazí se mu tlačítko koupit kartu. V souvislostí s tím máme metodu *BuyCard,* což je metoda pro koupi karty. Pokud má hráč finance na koupi karty, stane se jejím vlastníkem a znovu se vykreslí aktuální herní plocha, aby bylo vidět, že už je hráč vlastníkem karty a kolik peněz má k dispozici, jakmile dostatek financí nemá, vypíše se hláška, že si kartu nemůže finančně dovolit. Dále pokud je na tahu hráč, který kartu vlastní, zobrazí se mu tlačítko prodat kartu. Když kartu prodá, nastaví se vlastník na *null*, hráči jsou přičteny peníze a znovu se vykreslí aktuální herní plocha. Oproti kartám trenérů u karet koní vypisujeme na plochu také počet dostihů, které jsou na kartě a pokud je jich více než čtyři (tedy karta má hlavní dostih), vypíše se hlavní dostih. Když na koni žádné dostihy nejsou, nevypisujeme nic. Kdykoliv hráč, který je na tahu stojí na koni a vlastní všechny koně z této stáje, a ještě nemá zakoupen hlavní dostih, zobrazí se mu tlačítko pro koupi dostihů. Na koupi dostihů používáme obdobnou metodu jako u nákupu karty s tím rozdílem, že se ptáme, zdali již nemá hráč všechny dostihy zakoupené. A stejný princip je i u prodeje dostihů, kdy může dostihy prodat, kdykoli je na tahu.

Na úvodní obrazovce máme tlačítko *Nová hr*a a také si zde volíme počet hráčů ve hře pomocí *ComboBoxu* od dvou do čtyř. Poté co hráč klikne na tlačítko *Nová hra* se zavolají metody *CreateGame* pro přípravu hry a metoda *DrawBoard* pro vykreslení hrací plochy.

Na konci hry se zobrazí okno s pořadím hráčů a kolik peněz jednotliví hráči na konci hry měli k dispozici.

Veškerou aktivitu v tomto projektu řídí třída *UIService.cs*. Třída má hlavní metodu *DrawBoard*, která připraví hrací plochu, všechny karty koní, trenérů a další. Jedno políčko na hrací ploše je tvořeno ze dvou *StackPanelů* umístěných pod sebe. V horním *StackPanelu* je zobrazena karta a ve spodním jsou zobrazeny figurky hráčů. Metoda *DrawBoard* vždy při zavolání smaže veškeré věci z *gridu*. Při obětovném volání metody *DrawBoard* je nezbytné, aby byl *grid* čistý. Další, co jsem zde řešil, bylo přidání kostky a přidání tlačítka pro předání tahu. Obrázku kostky i tlačítku předání tahu přidáme *Tag*, který se používá pro ukládání objektů. A poté využívá objekty v *metodách MovePlayerByDice* a *PassTurn*.

```
if (!game.PlayerOnTurn.DiceRolled)
{
    //Přidání kostky
    StackPanel spDice = new StackPanel { HorizontalAlignment = HorizontalAlignment.Center, VerticalAlignment = VerticalAlignment.Center };

    var imgIconDice = new Image { HorizontalAlignment = HorizontalAlignment.Center, VerticalAlignment = VerticalAlignment.Center, Source = ne
    spDice.Children.Add(imgIconDice);

    //Přiřazení tagu obrázku kostky, abychom mohli využít game a grid v metodě MovePlayerByDice
    //Tag se používá pro ukládání objektu
    imgIconDice.Tag = new { Game = game, Grid = grid };
    //Přidání metody po kliknuhí na obrázek kostky
    imgIconDice.MouseDown += MovePlayerByDice;

    Grid.SetColumn(spDice, 4);
    Grid.SetRow(spDice, 4);
    grid.Children.Add(spDice);
}
else
{
    //Přidání tlačítka pro předání tahu
    var buttonNextPlayer = new Button { Content = "PŘEDEJTE TAH", HorizontalAlignment = HorizontalAlignment.Center, VerticalAlignment = Verti
    buttonNextPlayer.Padding = new Thickness(8);

    Grid.SetColumn(buttonNextPlayer, 4);
    Grid.SetRow(buttonNextPlayer, 4);
    grid.Children.Add(buttonNextPlayer);

    //Přiřazení tagu tlačítku, abychom mohli využít game a grid v metodě PassTurn
    buttonNextPlayer.Tag = new { Game = game, Grid = grid };
    //Přidání metody po kliknuhí na tlačítko
    buttonNextPlayer.Click += PassTurn;
}
```

Obrázek 3- Přidání kostky a tlačítka pro předání tahu

Dále je zde přidání aktuálního hráče, přičemž se vedle kostky vypíše, kdo je aktuálně na tahu a kolik má daný hráč peněz. Statistiky hráčů se zobrazují na levé straně obrazovky a obsahují jméno hráče, dostupné finance, počet vlastněných koní a trenérů a také jestli je stále ve hře nebo nikoli. V další metodě se přidávaly karty. Nejdříve karta startu a karta parkoviště, kde se pouze zvolila barva, ohraničení a pozice na desce. Karty trenérů a koní byly už složitější záležitost. U karet koní, je barva stejná jako barva stáje, ve které se nachází. U koní se také zobrazují dostihy. Dostih první až čtvrtý je žlutý a zobrazuje se společně s počtem dostihů. Hlavní dostih je červený a zobrazuje se s číslem pět. Pokud je to karta trenéra, nastavíme barvu *StackPanelu* na šedou a barvu nápisu na bílou. Stejně jako u kostky přidáme *Tag* obrázku ať už koně nebo trenéra, abychom mohli využít objekty v metodě *ShowCardInformation.* Zároveň pokud má karta vlastníka, vykreslíme barvu hráče okolo karty a přidáme tloušťku ohraničení. A naposledy v metodě *DrawBoard* přidáváme aktivní hráče ve hře na jejích pozice na ploše. Toto vykreslování děláme pomocí listu hráčů, kde si vezmeme pozici každého z nich a v důsledku toho je vykreslíme na plochu. Když je hráč na tahu, na poli, na kterém se nachází zabarvíme *StackPanel* zeleně, aby se mohl lépe orientovat na hrací desce.

Druhá metoda v této části je metoda *MovePlayerByDice,* která řeší posunutí hráče po kliknutí na kostku. Nejdříve vezmeme informace z tagu, hodíme kostkou, vypíšeme, kolik hráč hodil, posuneme ho na hrací ploše (nastavíme mu nový index a lokaci) a *booleanu DiceRolled* přiřadíme hodnotu *true*, aby hráč nemohl házet vícekrát. Dále pokud hráč prošel startem, přidáme mu peníze a vypíšeme hlášku, že prošel startem. Také musíme zjistit na jaké kartě hráč stojí a pokud je vlastník jiný hráč zaplatíme částku podle toho, kolik má dostihů na koni anebo kolik má zakoupených trenérů. Poté zavoláme metodu *DrawBoard* a vykreslíme hrací plochu s aktuálními informacemi. A nakonec zjistíme, zda hráč nezkrachoval, pokud ano, vyřadíme ho ze hry a všechny jeho karty vrátíme zpět do hry jako nezakoupené a znovu vykreslíme plochu. Jakmile hráč zkrachuje nastavíme hráči konečné pořadí podle toho, kolik je ve hře aktivních hráčů a jeho zneaktivníme. A také zjistíme, zdali se ve hře nachází poslední aktivní hráč, pokud ano, zobrazíme okno s koncem pořadím hráčů.

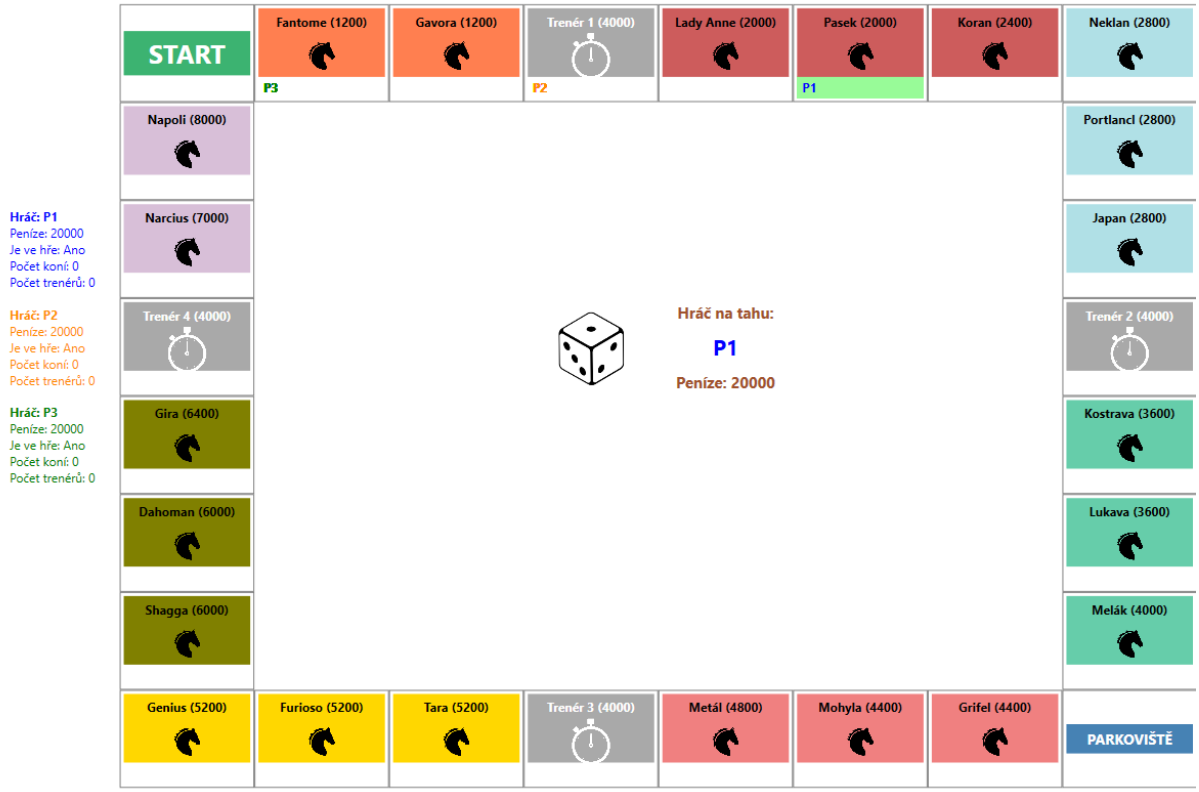
```
//Zkontrolujeme, jestli hráč nezkrachoval, pokud ano, vyřadíme ho ze hry a všechny jeho karty vrátíme do hry
if (game.PlayerOnTurn.Cash < 0)
{
    MessageBox.Show("Prohrál jste!", "Prohra hráče.");
    for (var i = 0; i < game.Board.Count; i++)
    {
        if (game.Board[i].Owner == game.PlayerOnTurn)
        {
            game.Board[i].Owner = null;
            if (game.Board[i].GetType() == typeof(HorseCard))
            {
                //Nezapomeneme nastavit hodnoty dostihů na výchozí hodnoty
                var horse = (HorseCard)game.Board[i];
                horse.NumberOfBets = 0;
            }
        }
    }

    //Nastavíme hráči konečné pořadí podle toho, kolik je ve hře aktivních hráčů a jeho zneaktivníme
    game.PlayerOnTurn.BoardIndex = -(game.Players.Where(x => x.IsActive).Count());
    game.PlayerOnTurn.IsActive = false;

    //Zjistíme, zda-li se ve hře nachází poslední aktivní hráč, pokud ano, zobrazíme okno s koncem pořadím hráčů
    if (_gameservice.IsGameOver(game))
    {
        game.Status = BettingOnHorsesModel.Enums.GameStatus.over;
        //Vykreslíme aktuální hrací plochu
        DrawBoard(game, ref grid);

        var endofgame = new EndOfGame(game);
        endofgame.ShowDialog();
        return;
    }

    //Vykreslíme aktuální hrací plochu
    DrawBoard(game, ref grid);
}
```

Obrázek 4- Vyřazení hráče ze hry

A poslední metoda se jmenuje *PassTurn*, což je metoda, zavolaná kliknutím na tlačítko předat tah. Předá tak dalšímu aktivnímu hráči a znovu vykreslí hrací plochu s aktuálními informacemi.

3. Závěr

Se svou ročníkovou prací jsem spokojen. Hra má většinu vlastností, co originální hra a sám jsem práci pojal jako vytvoření vlastní verze, které je v některých krocích jednodušší. Také jsem rád, že se mi podařilo vyřešit složitější problémy ve hře jako je například kupování dostihů na jednotlivých koní.



Obrázek 5- Hrací deska

4. Seznam obrázků

5. Bibliografie

Chand, M. (2020, Březen 7). Retrieved from C#Corner: https://www.c-sharpcorner.com/uploadfile/mahesh/combobox-in-C-Sharp/#:~:text=C%23%20ComboBox%20is%20a%20combination,in%20a%20drop%20down%20list.

Documentation. (n.d.). Retrieved from Microsoft: https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.count?view=netcore-3.1

Febrianto, B. (n.d.). *Money Icon Illustration*. Retrieved from UPlabs: https://www.uplabs.com/posts/money-icon-illustration-2363bbf0-a423-4e6d-85d4-13da33e21821

Freepik. (n.d.). *Free Icon*. Retrieved from Flaticon: https://www.flaticon.com/free-icon/horse-head_32610?term=horse&page=1&position=30&page=1&position=30&related_id=32610&origin=tag

Freepik. (n.d.). *Free Icon*. Retrieved from Flaticon: https://www.flaticon.com/free-icon/black-head-horse-side-view-with-horsehair_32904?term=horse%20head&page=1&position=31&page=1&position=31&related_id=32904&origin=tag

ho, a. (n.d.). *Icons*. Retrieved from Noun Project: https://thenounproject.com/term/white-dice/2032531/

Mooms. (n.d.). *Icons*. Retrieved from Noun Project: https://thenounproject.com/term/running-shoe/1473807/

Nyno, K. (n.d.). *Stopwatch*. Retrieved from ClipArtKey: https://www.clipartkey.com/view/biJiio_stopwatch-stopwatch-clipart/

Panels. (n.d.). Retrieved from WPF Tutorial: https://www.wpf-tutorial.com/panels/stackpanel/

Panels. (n.d.). Retrieved from WPF Tutorial: https://www.wpf-tutorial.com/panels/grid/

Panels. (n.d.). Retrieved from WPF Tutorial: https://www.wpf-tutorial.com/panels/grid-rows-and-columns/

Polyvore. (n.d.). *Red circle*. Retrieved from Clipartbest: http://www.clipartbest.com/clipart-yikgRMBpT

Tag. (n.d.). Retrieved from DotNetPerls: https://www.dotnetperls.com/tag

wpf. (n.d.). Retrieved from Tutorialspoint: https://www.tutorialspoint.com/wpf/wpf_xaml_overview.htm#:~:text=XAML%20stands%20for%20Extensible%20Application,of%20objects%20with%20hierarchical%20relations.