

# Dokumentáció



Tartalom:

 [Felhasználói kézikönyv](#)

 [Metódusok rövid leírása](#)




- Burger
- Component
  - Bun
  - Cheese
  - Lettuce
  - Onion
  - Patty
  - Pickle
  - Tomato
- Customer
- Day
- FileHandler
  - SaveToFile
  - LoadFromFile
- Game
- Order
- UserInterface
  - Window
  - MenuUI
    - CenterPanel
    - TopPanel
  - LoadGameUI
    - LoadBtn
    - EmptySaveBtn
    - GridPanel
  - ContinueBtn
  - FinishBtn
  - EndOfDayUI
  - Customer
    - GradeBurgerUI
    - CustomerOrderUI
  - CookingStationUI
  - BurgerDisplay
  - BurgerAssembleUI

 [Osztálydiagram](#)

Amennyiben az UML nem látszik rendesen, [itt](#) található jó minőségben.

## Felhasználói kézikönyv

A program elindításakor a felhasználó a Menüt láthatja maga előtt. Itt 3 gomb jelenik meg előtte:

-  Load Game gombra kattintva a program átirányítja a felhasználót a mentések betöltésére szolgáló oldalra (Load Game)
-  New Game gombra kattintva új játékot kezd a játékos
-  Exit segítségével kiléphet a játékból

A Load Game ablakon találhatóak a korábbi mentések. Egyszerre legfeljebb 3 mentést tárol a program, amennyiben a felhasználó új játékot kezd és mind a 3 mentési hely tele van, a játék a legrégebben frissített mentést írja felül. A felhasználó a mentéshez megfelelő gombra kattintva töltheti azt be (folytathatja a mentés által tartalmazott játékot).

Egy játék (meglévő vagy új) elindításakor a játékos láthatja, hogy hányadik nap az aktuális. Ez számára azért lehet releváns, mert a napok számával arányosan növekszik a vendégek száma illetve nehézsége egy nap.

A „continue” gombra kattintva a rendelés képernyőre navigálódik a játékos, ahol az aktuális vendég leírja neki a rendelését, ezt a felhasználónak ajánlott megjegyeznie, ugyanis ezt emlékezetből kell majd rekreálnia.

Ezután egy sütőlapot, illetve egy húspogácsát lát a játékos, a sütőn lévő csúszkával állíthatja be a hőmérsékletet, valamint a szövegmezőbe írhatja percekben, hogy meddig kívánja sütni a burgert. Amennyiben mindent beállított, a „cook” gombra kattintva elvégezheti a sütést.

A következő képernyőn láthat egy félkész burgert a húspogácsával, melyet az imént süített meg, illetve egy legördülő listát, melyben feltétek találhatók. A lista alatti gomb megnyomásával adhatja hozzá a burgerhez a feltétet. Ha készen van az összerakással, tovább mehet a „continue” gombbal.

Ekkor a vendég visszatér és borraivalót ad az alapján, hogy mennyire sikerült teljesíteni a rendelést.

Ezután a vásárlók sorra jönnek tovább, amíg már nincs több az adott napra. Amikor ez bekövetkezik, egy összesítő képernyőre kerül a játékos, ahol láthatja mennyi volt a profit (összes borraivaló) az adott napra, valamint az egyenlegét. Ha itt tovább lép, a következő nap következik. Ez addig folytatódik, amíg a játékos ki nem lép a játékból.

A program automatikusan menti a jelenlegi játékot. Egyszer amikor az elindul, illetve minden nap elején.

## Metódusok rövid leírása

A lista első szintjén látható hogy melyik class metódusairól beszélünk. Ahol `<Classname>.<Classname>` formátumban van megadva, az azt jelenti, hogy a második névvel rendelkező class származik az elsőből.

### **BURGER**

- `public Burger()` – inicializál egy új burgert, illetve hozzáad egy alsó bucit.
- `public getComponents()` – visszaadja a burger komponenseinek listáját.
- `public addComponent(Component c)` – a paraméterként kapott komponenst hozzáadja a burger komponenseinek listájához.
- `public setComponents(List<Component> c)` – beállítja a paraméterként kapott Component listát a burger komponens listájának

### **COMPONENT**

- `public getImg()` – visszaadja a komponens képét
- `public abstract toString()` – absztrakt metódus, minden komponens overrideolja

### **COMPONENT.BUN**

- `public Bun(boolean top)` – konstruktor, a top paraméter megadja, hogy felső vagy alsó buci, ez alapján állítja be az objektum képét is.
- `public isTop()` – visszaadja, hogy alsó vagy felső bun
- `public toString()` -a Component osztály által definiált metódus

### **COMPONENT.CHEESE**

- `public Cheese()` – konstruktor
- `public toString()` -a Component osztály által definiált metódus

### **COMPONENT.LETTUCE**

- `public Lettuce()` – konstruktor
- `public toString()` -a Component osztály által definiált metódus

### **COMPONENT.ONION**

- `public Onion()` – konstruktor
- `public toString()` -a Component osztály által definiált metódus

### **COMPONENT.PICKLE()**

- `public Pickle()` – konstruktor
- `public toString()` -a Component osztály által definiált metódus

### **COMPONENT.TOMATO()**

- `public Tomato()` – konstruktor
- `public toString()` -a Component osztály által definiált metódus

### **COMPONENT.PATTY()**

- `public Patty()` – konstruktor
- `public getCookingLevel()` – visszaadja, hogy mennyire van megsütve a patty
- `public setCookingLevel(int cookingLevel)` – beállítja a patty sütési szintjét a paraméterben megadott értékre
- `public getCookingLevelAsString()` – visszaadja a patty sütési szintjét szöveges formában
- `public getCookingLevelBasedOnHeatAndTime(int heat, int time)` – visszaadja a patty sütési szintjét a paraméterekként megadott hőmérséklet és idő értékének függvényében.
- `public toString()` -a Component osztály által definiált metódus

### **CUSTOMER**

- `public Customer(int d)` – konstruktor, a paraméterben megadott nehézséggel inicializálja az objektumot.
- `public getOrder()` – visszaadja a vásárló rendelését.
- `public getDifficulty()` – visszaadja a vásárló nehézségi szintjét.
- `public getImage()` – visszaadja a vásárló képét.
- `private randomiseImageBasedOnDifficulty()` – a nehézség alapján generál egy képet a vásárlónak
- `private generateEasyImage()` – beállítja a vásárló képét, a könnyű vásárlók közül választ.
- `private generateNormalImage()` – beállítja a vásárló képét, a közepes vásárlók közül választ.
- `private generateHardImage()` – beállítja a vásárló képét, a nehéz vásárlók közül választ.

### **DAY**

- `public Day(int count)` – inicializálja az adott napot, a paraméter arra vonatkozik, hogy hanyadik nap ez.
- `public generateCustomer()` – hozzáad a napi vásárlók listájához egy új vásárlót. A vásárló nehézsége a nap számától függ és randomizált. Amennyiben vége a napnak (a vásárlók száma elérte a nap számát), ezt jelzi.
- `public isDone()` – megmondja, hogy vége van-e már a napnak.
- `public addBurger(Burger burger)` – a napi burgerek listájához adja a paraméterként megadott burgert.
- `public increaseIncome(int amount)` – megnöveli a napi bevételt a paraméterként kapott összeggel.
- `private addCustomer(Customer customer)` – hozzáad egy vásárlót a napi vásárlók listájához.
- `public getCustomers()` – visszaadja a vásárlók listáját.
- `public getCurrentCustomer` – visszaadja a jelenlegi vásárlót (`currentCustomer`).

- `public setCurrentCustomer (Customer currentCustomer)` – beállítja a jelenlegi vásárlót.
- `public getBurgers()` – visszaadja a napi burgerek listáját
- `public getCustomer(int i)` – visszaadja az *i* indexnek megfelelő vásárlót a listából.
- `public getCount()` – visszaadja a nap számát.
- `public getProfit()` – visszaadja a napi bevételt.

#### **SAVEToFile**

- `private static serializeGame(Game game, int slot)` – szerializálja a megadott játékot a szintén megadott slot-ba (vagyis a slot-al megegyező számú save file-ba)
- `public static saveGameToFile(Game game)` – elmenti az adott játékot egy új fájlba. Ha mindhárom slot betelt, törli a legrégebben mentett sloton lévő mentést.
- `public static saveGameToAlreadyExistingFile(Game game, int slot)` – elmenti a játékot egy már létező slotra.
- `public static getSlotOfFileLeastRecentlyEdited()` – visszaadja a legrégebben frissített mentés slotjának számát.
- `private static getSaveFileLeastRecentlyEdited()` – visszaadja a legrégebben frissített fájlt.

#### **LOADFromFile**

- `public static loadFileFromSlot(int slotNumber)` – visszaadja a paraméterben megadott slotban lévő mentést.
- `public static loadGameFromSlot(int slotNumber)` – visszaadja a paraméterben megadott sloton tárolt játék objektumot.

#### **GAME**

- `public Game()` – default konstruktor
- `public Game(List<Day> days, int money, List<Upgrades> upgradesList)` – paraméteres konstruktor
- `public getDays()` -visszaadja a napok listáját
- `public getToday()` – visszaadja az aktuális napot.
- `public getDay()` – visszaadja hogy hanyadik nap van
- `public getPastDay(int d)` – visszaadja a paraméterben kapott *d* indexen található napot a listában
- `public newDay()` – hozzáad egy új napot a napok listájához
- `public getMoney()` – visszaadja a játék során szerzett pénzt.
- `public setMoney(int money)` – beállítja a pénzt az adott értékre.
- `public getUpgradesList` – visszaadja a fejlesztések listáját – **nem aktuális**
- `public addUpgrade(Upgrade upgrade)` – hozzáad egy új fejlesztést a listához - **nem aktuális**

- `public increaseMoney(int amount)` – megnöveli a játék során szerzett pénzt a paraméterként kapott mennyiséggel.
- `public decreaseMoney(int amount)` – csökkenti a játék során szerzett pénzt a paraméterként kapott mennyiséggel.
- `public increaseMoneyBasedOnScore(int score)` – megnöveli a játék során szerzett pénzt a paraméterként megadott pontszám függvényében.
- `public toString()` – visszaadja az objektum adatait String formátumban.

#### ORDER

- `public Order(int difficulty)` – konstruktor
- `private generateBurger()` – generál egy burgert, amelyet nehézség alapján 2-4-6 véletlenszerű komponenssel tölt fel
- `private fillOrder(Burger b, int h)` – feltölti a paraméterként megadott burgert annyi random komponenssel, amennyi a h értéke.
- `public compareBurgerToOrder(Burger b)` – összehasonlítja a rendelésben szereplő burger komponenseit a paraméterben kapott burger komponenseivel és visszaadja az egyező komponensek számát.
- `public getBurger()` – visszaadja a burgert.
- `public getDifficulty()` – visszaadja a nehézséget.

### UserInterface osztályai:

#### WINDOW

- `public Window()` – konstruktor
- `public setup()` – beállítja a komponensek és az ablak méretét.
- `public getWidth()` – visszaadja az ablak szélességét.
- `public getHeight()` – visszaadja az ablak magasságát.
- `public addCards()` – a `cardLayout`-hoz adja hozzá a különböző képernyőkhöz tartozó komponenseket.
- `finishCooking(Patty p, int h, int t)` – sültés fázis befejezése, továbbadja az adatokat az `assembly`-nek és elvégzi a tényleges sütést is a húspogácsán.
- `finishAssembling(Burger b)` – `assembly` fázis befejezése. Az összerakott burgert továbbadja a `GradeBurger`-nek.
- `showScreen(String n)` – megjeleníti az n névhez tartozó kártyát.
- `startGame(Game game, int slot)` – elindítja a kapott játékot és el is menti azt a megadott slotra.
- `nextDay()` – elindítja a következő napot.
- `nextCustomer()` – áttér a következő vendégre. Ha nincs több vendég az adott nap, akkor meghívja a `nextDay`-t.
- `showNextScreen()` – a `continueBtn`-ek ezt hívják meg kattintáskor, az aktuális képernyő alapján vált a következőre.
- `getGame()` – visszaadja a jelenleg futó játékot.

### MENUUI

- public MenuUI(Window window) – konstruktor
- private newBtnSetup(Window w) – a newBtn komponenst állítja be és adja hozzá a menühöz
- private exitBtnSetup() – az exitBtn komponenst állítja be és adja hozzá a menühöz
- private loadBtnSetup(Window w) – ugyanez loadBtn-el
- private centerPanelSetup(Window w) – ugyanez centerPanel-el, illetve hozzáadja a gombokat.
- protected paintComponent(Graphics g) – háttérképet állítja
- private void setup(Window window) – beállítja a komponenseket és a menü komponens tulajdonságait

### CENTERPANEL

- public CenterPanel() - konstruktor

### TOPPANEL

- public TopPanel() - konstruktor

### LOADGAMEUI

- public LoadGameUI() – konstruktor
- private loadBtnSetup() – beállítja a save fileok számának és tartalmának függvényének hogy hány loadBtn és hány EmptySaveBtn legyen

### LOADBTN

- public LoadBtn(int s) – s a slotNumber ahol a gombhoz tartozó játékot tároljuk.
- public getGame() – visszaadja a játékot ami a megfelelő slotban van.
- public getSlot() – visszaadja a slot értékét.

### EMPTYSAVEBTN

- public EmptySaveBtn() – konstruktor - inicializálja a gombot és tartalmát.

### GRIDPANEL

- public GridPanel() - konstruktor

### CONTINUEBTN

- public ContinueBtn(Window w) – konstruktor

### FINISHBTN – **nem aktuális**, elfelejtettem használni

### ENDOFDAYUI

- public EndOfDayUI(Window w) – konstruktor
- public update() – frissíti a szövegmezők értékét és újra festi a panelt
- protected paintComponent(Graphics g) – háttér beállítását végzi

### **GRADEBURGERUI**

- `public GradeBurgerUI(Window w, int p)` – konstruktor, `p` a borralaló értéke
- `protected labelSetup()` – a profit label értékeit állítja be
- `protected burgerSetup()` – a burger komponenst állítja fel
- `protected pultSetup()` – a pultot állítja fel
- `protected customerSetup()` a vásárló képét jeleníti meg
- `protected void paintComponent(Graphics g)` – a komponens háttérképét állítja

### **CUSTOMERORDERUI**

- `public CustomerOrderUI(Window w)` – konstruktor
- `protected void initComponents()` – beállítja a komponensek méreteit, értékét, majd hozzáadja azokat a panelhez.
- `protected pultSetup()` – a pultot állítja fel
- `protected customerSetup()` – a vendég képét jeleníti meg
- `protected void setupContinueBtn` – a continue btn-t állítja be
- `public void cycleToNextCustomer()` – frissíti az értékeket és újra rajzolja a komponenst
- `public newDay()` - frissíti az értékeket és újra rajzolja a komponenst
- `protected setupOrderList()` – a rendelés listát építi fel
- `protected makeOrderListStriped()` – becsíkozza az order listet
- `protected paintComponent(Graphics g)` – a háttérét állítja be

### **COOKINGSTATIONUI**

- `public CookingStationUI(Window w)` – konstruktor
- `protected cookingSetup()` – setupolja a komponenseket.
- `public reset()` – visszaállítja a csúszka és szövegmező értékeit eredetire
- `protected paintComponent(Graphics g)` – háttér

### **BURGERDISPLAYER**

- `public BurgerDisplayer(Patty p, Window w)` – konstruktor, ha új burgert akarunk megjeleníteni, majd esetleg elemeket hozzáadni
- `public BurgerDisplayer(Window w, Burger b)` – konstruktor, ha meglévő burgert akarunk megjeleníteni
- `protected addComponent(Component c)` – hozzáad egy komponenst a burgerhez és újrafesti a panelt.
- `protected setup(Window w)` – beállítja a komponenseket.
- `public getBurger()` – visszaadja a megjelenített burgert.

### **ASSEMBLEBURGERUI**

- `public AssembleBurgerUI(Window w, Patty p)` – konstruktor
- `protected loadComponents()` – betölti a komponens listát
- `protected paintComponent(Graphics g)` – háttér megjelenítéséért felelős

