# شرح تفصيلي لملفات Utils.cpp و Utils.h

## نظرة عامة

ملفات **Utils** (اختصار Utilities) هي مكتبة من الأدوات المساعدة التي تستخدمها أجزاء مختلفة من البرنامج. هذه الدوال "عامة" وليست مرتبطة بفئة معينة، مما يجعلها قابلة لإعادة الاستخدام في أي مشروع.

---

## ملف Utils.h (ملف الواجهة)

### 1. هيكل Puzzle

```cpp
struct Puzzle {
    string riddle;   // السؤال أو اللغز
    string answer;   // الإجابة الصحيحة
    string category; // الفئة (مثل: Math, Logic, General)
};
```

**الغرض**: تخزين معلومات لغز واحد بطريقة منظمة.

**مثال**:

```cpp
Puzzle p;
p.riddle = "What has keys but no locks?";
p.answer = "keyboard";
p.category = "Technology";
```

---

### 2. دوال معالجة النصوص

أ. **string trimText(const string& text)**

- **الوظيفة**: إزالة المسافات والأحرف الخاصة من بداية ونهاية النص
- **الاستخدام**: تنظيف إدخالات المستخدم
- **مثال**:

```cpp
```

```
trimText("   hello   ") → "hello"
trimText("\t\nworld\n") → "world"
```

**ب. `string toLowercase(const string& text)`**

- **الوظيفة**: تحويل جميع الأحرف إلى حروف صغيرة
- **الاستخدام**: المقارنات غير الحساسة لحالة الأحرف
- **مثال**:

```
cpp

toLowercase("HELLO World") → "hello world"
toLowercase("CamelCase") → "camelcase"
```

**ج. `string cleanInput(const string& input)`**

- **الوظيفة**: دمج trim و lowercase معاً
- **الاستخدام**: التحضير النهائي لمقارنة النصوص
- **مثال**:

```
cpp

cleanInput("  ANSWER  ") → "answer"
cleanInput("\t The Book \n") → "the book"
```

---

### 3. دوال تحميل الألغاز

**أ. `vector<Puzzle> loadPuzzlesFromFile(const string& filename)`**

- **الوظيفة**: قراءة ألغاز من ملف نصي وتحويلها إلى vector
- **القيمة المُرجعة**: مصفوفة ديناميكية من الألغاز
- **الاستخدام**: تحميل بنك الألغاز عند بدء البرنامج

**ب. `bool isPuzzleValid(const Puzzle& puzzle)`**

- **الوظيفة**: التحقق من أن اللغز يحتوي على سؤال وإجابة
- **القيمة المُرجعة**: `true` إذا كان اللغز صالح
- **الاستخدام**: فلترة الألغاز الفارغة أو المعطوبة

### 4. دوال عشوائية

أ. **`int randomNumber(int min, int max)`**

- **الوظيفة**: توليد رقم عشوائي ضمن نطاق محدد
- **المعاملات**: الحد الأدنى والأعلى (inclusive)
- **مثال**:

```cpp
randomNumber(1, 6)  → قد يعطي: 1, 2, 3, 4, 5, أو 6
randomNumber(10, 20) → رقم بين 10 و 20
```

ب. **`void shufflePuzzles(vector<Puzzle>& puzzles)`**

- **الوظيفة**: خلط ترتيب الألغاز بشكل عشوائي
- **الاستخدام**: تنويع الألغاز في كل مرة تُلعب فيها اللعبة
- **ملاحظة**: تُعدل على المصفوفة الأصلية (pass by reference)

---

## ملف Utils.cpp (التنفيذ)

### 1. دالة trimText()

**الخوارزمية:**

```cpp
```

```cpp
string trimText(const string& text) {
    // الخطوة 1: البحث عن أول حرف ليس مسافة
    size_t start = text.find_first_not_of(" \t\n\r");

    // الخطوة 2: إذا كل النص مسافات فقط
    if (start == string::npos) {
        return ""; // نرجع نص فارغ
    }

    // الخطوة 3: البحث عن آخر حرف ليس مسافة
    size_t end = text.find_last_not_of(" \t\n\r");

    // الخطوة 4: استخراج الجزء المطلوب
    return text.substr(start, end - start + 1);
}
```

**أمثلة تفصيلية:**

```
Input: "   hello world   "
       0123456789012345678

start = 3  (أول 'h')
end = 13   (آخر 'd')
length = 13 - 3 + 1 = 11
Output: "hello world"
```

**الأحرف المزالة:**

- `' '` مسافة عادية
- `'\t'` Tab
- `'\n'` سطر جديد (New Line)
- `'\r'` عودة إلى أول السطر (Carriage Return)

---

### 2. دالة toLowercase()

```
cpp
```

```cpp
string toLowercase(const string& text) {
    string result = text;  // نسخة من النص الأصلي

    // نمر على كل حرف
    for (int i = 0; i < result.length(); i++) {
        result[i] = tolower(result[i]);  // تحويل للحرف الصغير
    }

    return result;
}
```

**كيف يعمل `tolower()`؟**

- دالة من مكتبة `<cctype>`
- تحول الحرف الكبير إلى صغير
- إذا كان الحرف صغيراً أصلاً، يبقى كما هو
- الأرقام والرموز لا تتأثر

**مثال:**

```
'A' → 'a'
'Z' → 'z'
'a' → 'a' (بدون تغيير)
'5' → '5' (بدون تغيير)
'!' → '!' (بدون تغيير)
```

---

### 3. دالة cleanInput()

```cpp
cpp

string cleanInput(const string& input) {
    string trimmed = trimText(input);      // الخطوة 1: إزالة المسافات
    return toLowercase(trimmed);           // الخطوة 2: تحويل لحروف صغيرة
}
```

**مثال شامل:**

```
Input: "   THE ANSWER   "
↓ trimText()
"THE ANSWER"
↓ toLowercase()
"the answer"
```

**الفائدة**: جعل المقارنة بين إجابات اللاعب والإجابة الصحيحة عادلة:

```cpp
cleanInput("Keyboard") == cleanInput("keyboard")     → true
cleanInput("  THE BOOK  ") == cleanInput("the book") → true
```

---

### 4. دالة loadPuzzlesFromFile()

#### أ. فتح الملف

```cpp
vector<Puzzle> puzzles;
ifstream file(filename);

if (!file.is_open()) {
    cout << "Error: Can't open file " << filename << endl;
    return puzzles;  // نرجع vector فاضي
}
```

#### ب. صيغة الملف المتوقعة

```
[Category Name]
Riddle: What has keys but no locks? | Answer: keyboard
Riddle: I'm tall when young, short when old | Answer: candle

[Math]
Riddle: What is 2 + 2? | Answer: 4
```

#### ج. معالجة السطور

```cpp
```

```
string line;
string currentCategory = "General";  // الفئة الافتراضية

while (getline(file, line)) {
    line = trimText(line);

    // تخطي السطور الفارغة
    if (line.empty()) {
        continue;
    }

    // اكتشاف فئة جديدة: [Category]
    if (line[0] == '[' && line[line.length()-1] == ']') {
        currentCategory = line.substr(1, line.length()-2);
        continue;
    }

    // معالجة سطر اللغز...
}
```

### د. استخراج اللغز

cpp

```
// البحث عن الكلمات المفتاحية
size_t riddlePos = line.find("Riddle: ");
size_t answerPos = line.find(" | Answer: ");

if (riddlePos != string::npos && answerPos != string::npos) {
    // استخراج السؤال
    string riddle = line.substr(riddlePos + 8, answerPos - (riddlePos + 8));
    riddle = trimText(riddle);

    // استخراج الإجابة
    string answer = line.substr(answerPos + 11);
    answer = trimText(answer);

    // إنشاء اللغز
    Puzzle puzzle;
    puzzle.riddle = riddle;
    puzzle.answer = answer;
    puzzle.category = currentCategory;

    puzzles.push_back(puzzle);
}
```

**مثال على الاستخراج:**

```
السطر: "Riddle: What has keys? | Answer: keyboard"
       0123456789...

riddlePos = 0
answerPos = 23

السؤال:
  start = 0 + 8 = 8
  length = 23 - 8 = 15
  riddle = "What has keys?"

الإجابة:
  start = 23 + 11 = 34
  answer = "keyboard"
```

**هـ. الختام**

```cpp
file.close();
cout << "Loaded " << puzzles.size() << " puzzles successfully!" << endl;
return puzzles;
```

---

### 5. دالة isPuzzleValid()

```cpp
bool isPuzzleValid(const Puzzle& puzzle) {
    return !puzzle.riddle.empty() && !puzzle.answer.empty();
}
```

**بسيطة وواضحة:**

- اللغز صحيح إذا كان السؤال والإجابة غير فارغين
- تُستخدم للتأكد من عدم إضافة ألغاز معطوبة

**مثال:**

```cpp
```

```
Puzzle p1 = {"What is it?", "answer", "General"};
isPuzzleValid(p1) → true

Puzzle p2 = {"", "answer", "General"};
isPuzzleValid(p2) → false (السؤال فارغ)

Puzzle p3 = {"Question?", "", "General"};
isPuzzleValid(p3) → false (الإجابة فارغة)
```

---

### 6. دالة randomNumber()

```cpp
int randomNumber(int min, int max) {
    static bool seeded = false;

    // Seed مرة واحدة فقط
    if (!seeded) {
        srand(time(0));  // استخدام الوقت الحالي كـ seed
        seeded = true;
    }

    // توليد رقم عشوائي
    return min + (rand() % (max - min + 1));
}
```

**الشرح التفصيلي:**

**أ. متغير static:**

```cpp
static bool seeded = false;
```

- يُنشأ مرة واحدة فقط ويبقى في الذاكرة
- يحتفظ بقيمته بين استدعاءات الدالة

**ب. Seeding:**

```cpp
srand(time(0));
```

- `time(0)` يعطي الوقت الحالي بالثواني
- `srand()` تهيئة مولد الأرقام العشوائية
- بدون seed، ستحصل على نفس الأرقام في كل مرة!

**ج. التوليد:**

```
cpp

rand() % (max - min + 1)
```

- `rand()` يعطي رقم من 0 إلى RAND_MAX
- `% (max - min + 1)` يحصر النطاق
- `+ min` يضبط البداية

**مثال:**

```
randomNumber(1, 6):
  max - min + 1 = 6 - 1 + 1 = 6
  rand() % 6 → يعطي: 0, 1, 2, 3, 4, أو 5
  + min (1) → يعطي: 1, 2, 3, 4, 5, أو 6
```

---

### 7. دالة shufflePuzzles()

```
cpp

void shufflePuzzles(vector<Puzzle>& puzzles) {
    // إنشاء random engine
    static random_device rd;       // مصدر عشوائية حقيقي
    static mt19937 gen(rd());      // Mersenne Twister generator

    // خلط العناصر
    shuffle(puzzles.begin(), puzzles.end(), gen);
}
```

**الفرق بين random_device و rand():**

| الخاصية | rand() | random_device |
|---|---|---|
| الجودة | pseudo-random | true random |
| السرعة | سريع | أبطأ قليلاً |
| الاستخدام | أرقام بسيطة | تطبيقات حساسة |

### لماذا static؟

```cpp
static random_device rd;
static mt19937 gen(rd());
```

- لتجنب إعادة التهيئة في كل مرة
- التهيئة عملية مكلفة نسبياً
- يكفي إنشاؤها مرة واحدة

### كيف يعمل shuffle()؟

```cpp
shuffle(puzzles.begin(), puzzles.end(), gen);
```

- `begin()` و `end()` نطاق العناصر
- `gen` مولد الأرقام العشوائية
- يُبدل مواقع العناصر بشكل عشوائي

**مثال:**

```
قبل الخلط:
puzzles = [P1, P2, P3, P4, P5]

بعد الخلط:
puzzles = [P3, P1, P5, P2, P4]
```

---

## استخدامات الدوال في اللعبة

### 1. في Game.cpp

```cpp
// مقارنة الإجابات
bool Game::checkAnswer(string playerAnswer, string correctAnswer) {
    return cleanInput(playerAnswer) == cleanInput(correctAnswer);
}
```

### 2. في Library.cpp (افتراضي)

```cpp
// تحميل الألغاز عند بناء المكتبة
Library::Library() {
    vector<Puzzle> allPuzzles = loadPuzzlesFromFile("puzzles.txt");

    // خلط الألغاز
    shufflePuzzles(allPuzzles);

    // توزيعها على الكتب
    for (int i = 0; i < allPuzzles.size(); i++) {
        if (isPuzzleValid(allPuzzles[i])) {
            // إضافة اللغز لكتاب معين...
        }
    }
}
```

### 3. اختيار ألغاز عشوائية

```cpp
// اختيار 5 ألغاز عشوائية من 100
vector<Puzzle> selected;
for (int i = 0; i < 5; i++) {
    int index = randomNumber(0, allPuzzles.size() - 1);
    selected.push_back(allPuzzles[index]);
}
```

---

## نصائح تقنية

### 1. const string& vs string

```cpp
string trimText(const string& text)  // ✓ أفضل
string trimText(string text)         // ✗ أبطأ (نسخ كامل)
```

- `const string&` تمرير بالمرجع (بدون نسخ)
- `const` تضمن عدم تعديل النص الأصلي

### 2. size_t vs int

```cpp
size_t start = text.find_first_not_of(" ");
```

- `size_t` نوع خاص للأحجام والفهارس
- دائماً موجب (unsigned)
- `string::npos` قيمة خاصة تعني "لم يُعثر عليه"

### 3. static في الدوال

```cpp
static bool seeded = false;
```

- يُنشأ مرة واحدة فقط
- يبقى في الذاكرة طوال عمر البرنامج
- مفيد للتهيئة الثقيلة

---

### مثال شامل للاستخدام

```cpp
```

```cpp
int main() {
    // تحميل الألغاز
    vector<Puzzle> puzzles = loadPuzzlesFromFile("puzzles.txt");
    cout << "Loaded: " << puzzles.size() << " puzzles\n";

    // خلطها
    shufflePuzzles(puzzles);

    // اختيار لغز عشوائي
    int index = randomNumber(0, puzzles.size() - 1);
    Puzzle current = puzzles[index];

    // التحقق من صحته
    if (!isPuzzleValid(current)) {
        cout << "Invalid puzzle!\n";
        return 1;
    }

    // عرضه
    cout << "Riddle: " << current.riddle << endl;

    // قراءة الإجابة
    string answer;
    getline(cin, answer);

    // المقارنة
    if (cleanInput(answer) == cleanInput(current.answer)) {
        cout << "Correct!\n";
    } else {
        cout << "Wrong! Answer was: " << current.answer << endl;
    }

    return 0;
}
```

---

### الخلاصة

ملفات **Utils** توفر:

**دوال النصوص:**

- ✓ تنظيف الإدخالات
- ✓ مقارنات عادلة

- ✓ معالجة آمنة

**دوال الملفات:**

- ✓ تحميل بيانات منظم
- ✓ دعم فئات متعددة
- ✓ معالجة أخطاء قوية

**دوال العشوائية:**

- ✓ أرقام عشوائية موثوقة
- ✓ خلط عناصر متقدم
- ✓ تنويع اللعب

**الميزة الأساسية!**: هذه الدوال قابلة لإعادة الاستخدام في أي مشروع، ليست حصرية على لعبة المكتبة