الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة المعلوماتية

تقرير مشروع

خوارزميات البحث الذكية

إعداد الطلاب

أحمد البغدادي - براء دركزلي - بشر جمعة

عمر الحكيم - محمد الحلاق

العام الدراسي

2023-2022

الأفكار الرئيسية:

1. الهدف هو الوصول إلى المنزل مع مراعاة الكلفة المرادة
2. يجب تمثيل الكلاسات التالية : (تم تمثيلها في لغة Java)

- Station – Path
- Transport – Bus – Taxi – Walk
- State
- Logic
- Game
- Main
- User Input

Station: يمثل المحطات بين الطرق

Path : يمثل الطريق بين محطتين

Transport: هو كلاس مجرد يحوي الواصفات المشتركة لدى وسائل النقل المتاحة

Bus-Taxi-Walk : الكلاسات الثلاثة ترث من الكلاس Transport ويضيف كل منهم الخصائص الذي يحتاجها (إن وجد)

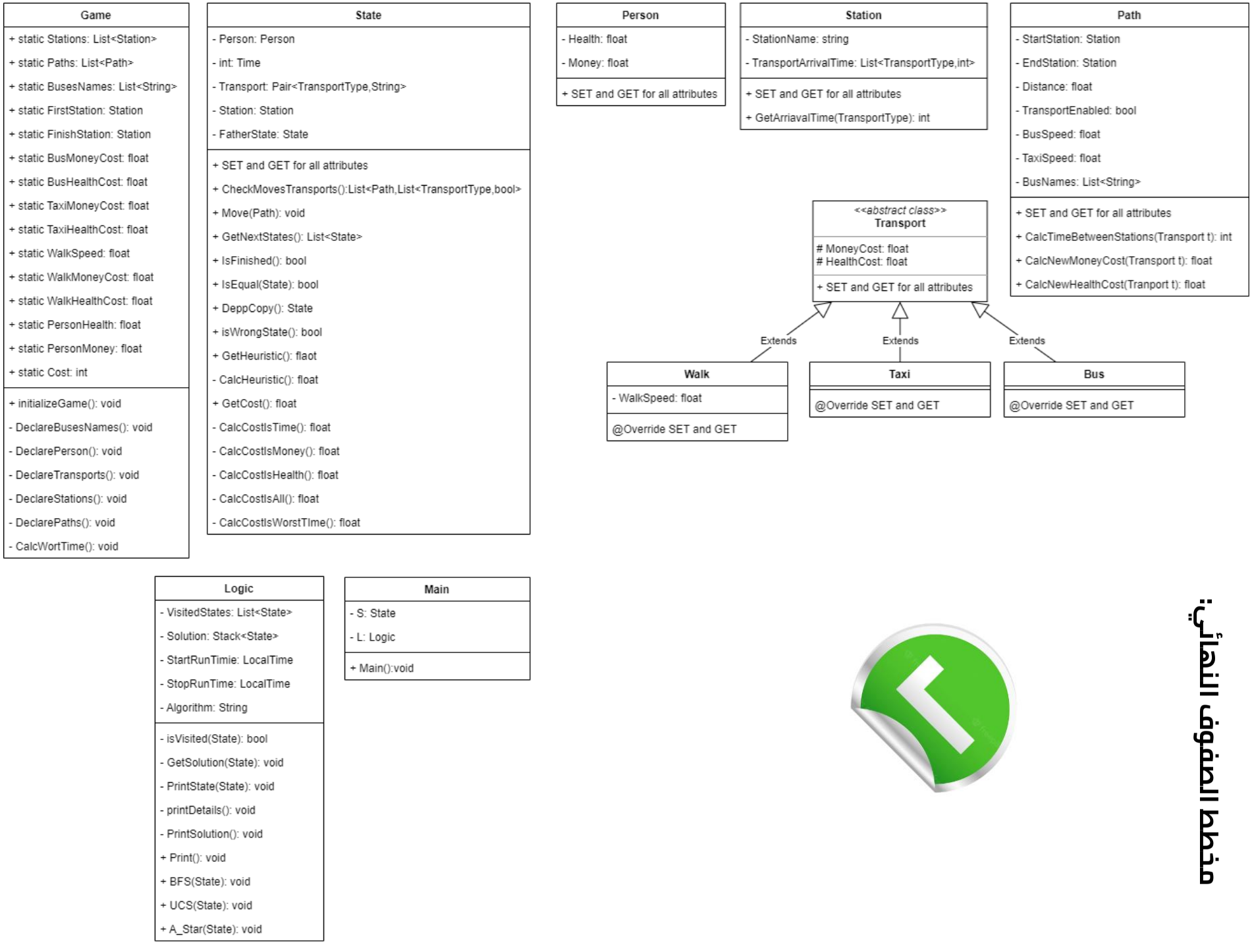
State : هو كلاس البنية التي تعبر عقدة في الGraph الذي سيبحث عن الحل وتحوي بداخلها بشكل أساسي توابع توصيف الحالة (Check Moves – Move – Get Next States – Is Equal – Is Finish)

Logic: يحوي توابع الخوارزميات المراد تطبيقها

Game: يحوي قيم ستاتيكية سهلة التعديل لتمثيل سيناريو معين وتجريب الخوارزمية عليه

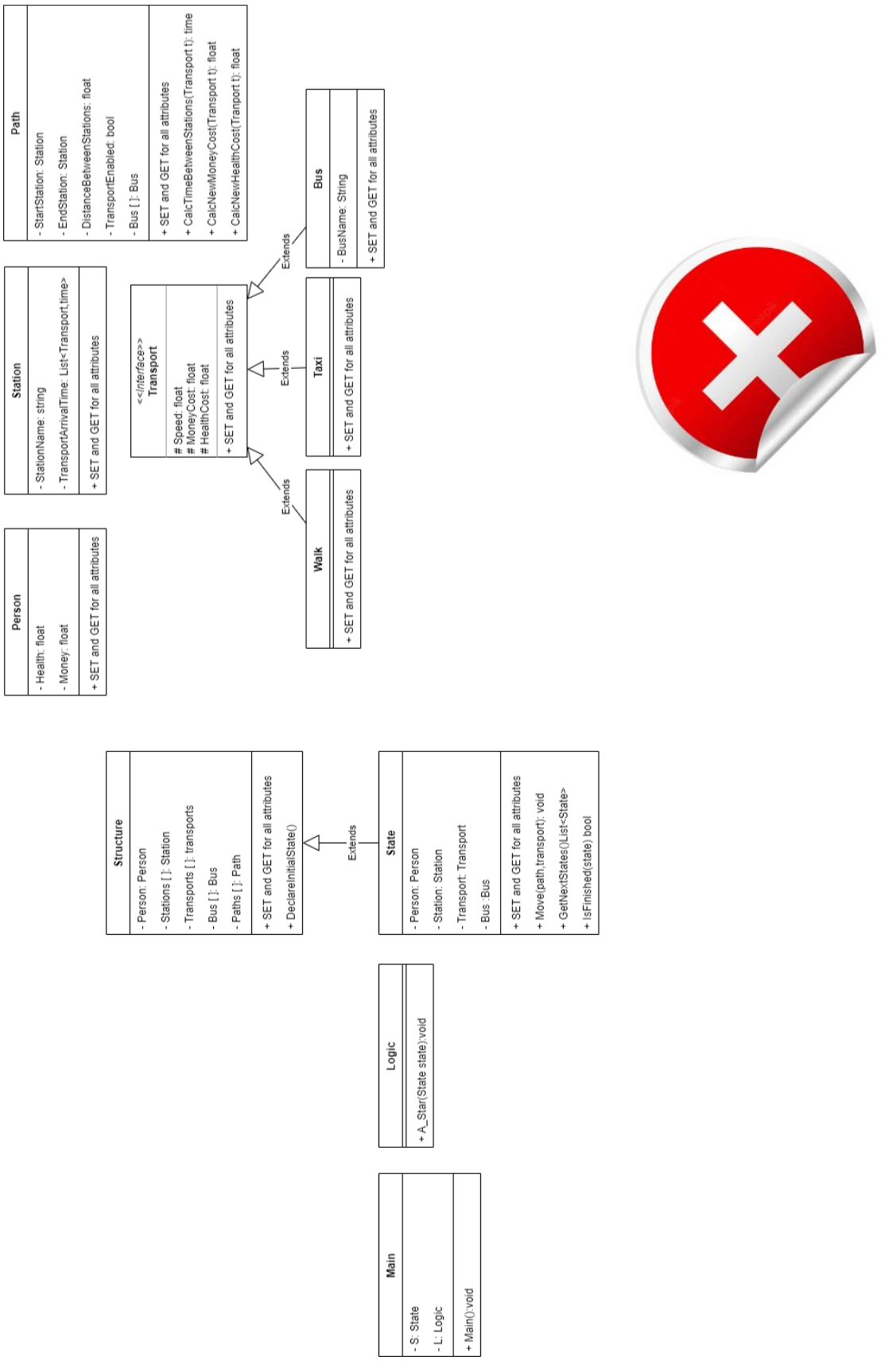
User Input: كلاس إضافي لإدخال البيانات من المستخدم

Main: الكلاس الرئيسي الذي يستدعي الخوارزمية

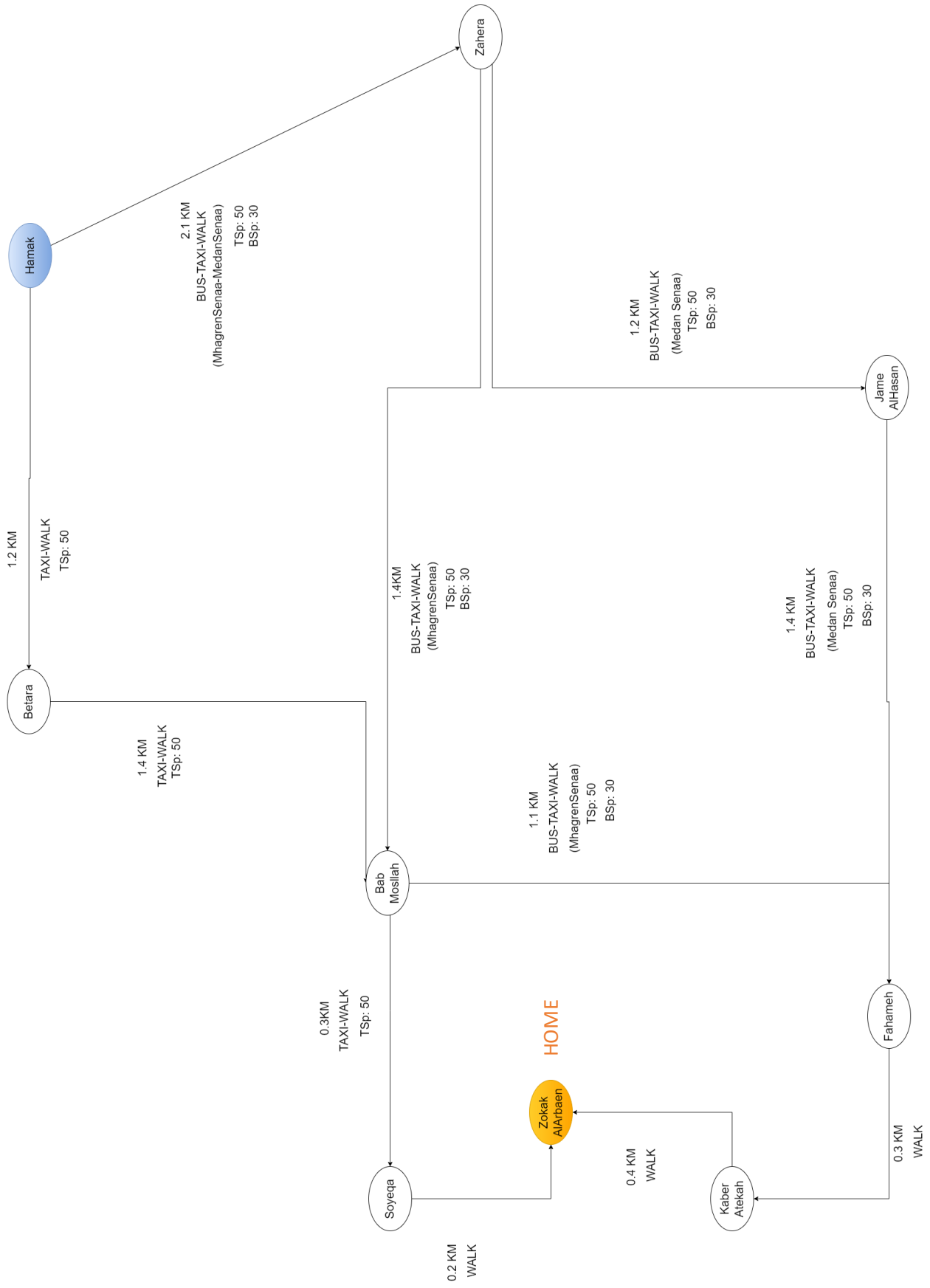


مخطط الصفوف النهائي:

مخطط الصفوف (النسخة الأولية التي تم الاتفاق عليها):



السيناريو الممثل في كلاس Game:



الحل:

بفرض أن الشخص كان يملك 3000 ليرة في جيبه

بفرض سرعة الباص 30 في جميع الطرق

بفرض سرعة التاكسي 50 في جميع الطرق

1. **الوصول إلى المنزل بأسرع وقت بحيث لا تتجاوز كلفة الطريق المبلغ الموجود في الجيب وبحيث لا تصل الطاقة إلى 0 عند نهاية الطريق الكلفة هي الزمن:**

```
no usages
public static void main(String[] args)  {
    Game.InitilizeGame();

    Game.SetCostTime();

    State S=new State();
    Logic L=new Logic();

    L.A_Star(S);
    L.print();
}

}
```

1

2

```
Algorithm: A*
Visited Count: 6
Steps Count: 4
Run Time: 00:00:00.001
Default Money Value: 3000.0
Cost By: Time
**********

Solution Path:

Hamak to AlBetara
Taxi
MoneyCost: -1200.0
HealthCost: 6.0
TimeCost: 2
TotalMoney: 1800.0
TotalHealth: 106.0
TotalTime: 2

AlBetara to BabMosalla
Taxi
MoneyCost: -1700.0
HealthCost: 8.5
TimeCost: 2
TotalMoney: 100.0
TotalHealth: 114.5
TotalTime: 4

BabMosalla to Soyika
Walk
MoneyCost: -0.0
HealthCost: -3.0
TimeCost: 3
TotalMoney: 100.0
TotalHealth: 111.5
TotalTime: 7

Soyika to ZokakAlArbaein
Walk
MoneyCost: -0.0
HealthCost: -2.0
TimeCost: 2
TotalMoney: 100.0
TotalHealth: 109.5
TotalTime: 9


Process finished with exit code 0
```

2. الوصول إلى منزلك بأقل كلفة ممكنة, بحيث ال تصل طاقتك إلى 0 عند نهاية الطريق

الكلفة هي المال:

```
    public static void main(String[] args)  {
        Game.InitilizeGame();

        Game.SetCostMoney();

        State S=new State();
        Logic L=new Logic();

        L.A_Star(S);
        L.print();
    }

}
```

1

2

```
Algorithm: A*
Visited Count: 5
Steps Count: 4
Run Time: 00:00:00.001
Default Money Value: 3000.0
Cost By: Money
**********************************


Solution Path:

Hamak to AlBetara
Walk
MoneyCost: -0.0
HealthCost: -12.0
TimeCost: 13
TotalMoney: 3000.0
TotalHealth: 88.0
TotalTime: 13

AlBetara to BabMosalla
Walk
MoneyCost: -0.0
HealthCost: -17.0
TimeCost: 19
TotalMoney: 3000.0
TotalHealth: 71.0
TotalTime: 32

BabMosalla to Soyika
Walk
MoneyCost: -0.0
HealthCost: -3.0
TimeCost: 3
TotalMoney: 3000.0
TotalHealth: 68.0
TotalTime: 35

Soyika to ZokakAlArbaein
Walk
MoneyCost: -0.0
HealthCost: -2.0
TimeCost: 2
TotalMoney: 3000.0
TotalHealth: 66.0
TotalTime: 37


Process finished with exit code 0
```

3. الوصول إلى منزلك بأعلى طاقة ممكنة, بحيث لا تتجاوز كلفة الطريق المبلغ الموجود في جيبك

الكلفة هي الطاقة:

```
    public static void main(String[] args)  {
        Game.InitilizeGame();

        Game.SetCostHealth();

        State S=new State();
        Logic L=new Logic();

        L.A_Star(S);
        L.print();
    }

}
```

1

```
Algorithm: A*
Visited Count: 6
Steps Count: 4
Run Time: 00:00
Default Money Value: 3000.0
Cost By: Health
**********

Solution Path:

Hamak to AlBetara
Taxi
MoneyCost: -1200.0
HealthCost: 6.0
TimeCost: 2
TotalMoney: 1800.0
TotalHealth: 106.0
TotalTime: 2

AlBetara to BabMosalla
Taxi
MoneyCost: -1700.0
HealthCost: 8.5
TimeCost: 2
TotalMoney: 100.0
TotalHealth: 114.5
TotalTime: 4
```

2

```
BabMosalla to Soyika
Walk
MoneyCost: -0.0
HealthCost: -3.0
TimeCost: 3
TotalMoney: 100.0
TotalHealth: 111.5
TotalTime: 7

Soyika to ZokakAlArbaein
Walk
MoneyCost: -0.0
HealthCost: -2.0
TimeCost: 2
TotalMoney: 100.0
TotalHealth: 109.5
TotalTime: 9


Process finished with exit code 0
```

4. **الوصول إلى منزلك بأسرع وقت ممكن، وبأقل كلفة ممكنة وبأعلى طاقة ممكنة**

الكلفة هي الزمن والمال والطاقة

```
public static void main(String[] args)  {
    Game.InitilizeGame();

    Game.SetCostAll();

    State S=new State();
    Logic L=new Logic();

    L.A_Star(S);
    L.print();
}

}
```

1

```
Algorithm: A*
Visited Count: 14
Steps Count: 4
Run Time: 00:00:00.001
Default Money Value: 3000.0
Cost By: All
*****************************

Solution Path:

Hamak to Zahera
Bus Mhagren Senaa
MoneyCost: -400.0
HealthCost: -10.5
TimeCost: 19
TotalMoney: 2600.0
TotalHealth: 89.5
TotalTime: 19

Zahera to BabMosalla
Bus Mhagren Senaa
MoneyCost: -0.0
HealthCost: -7.0
TimeCost: 3
TotalMoney: 2600.0
TotalHealth: 82.5
TotalTime: 22
```

2

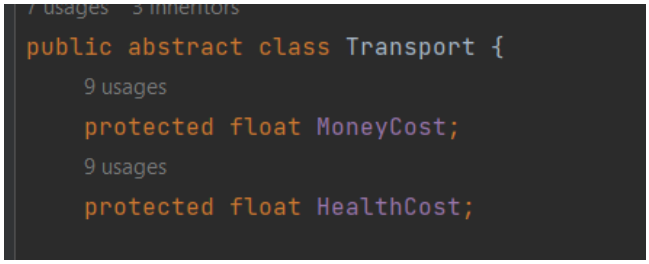
```
BabMosalla to Soyika
Walk
MoneyCost: -0.0
HealthCost: -3.0
TimeCost: 3
TotalMoney: 2600.0
TotalHealth: 79.5
TotalTime: 25

Soyika to ZokakAlArbaein
Walk
MoneyCost: -0.0
HealthCost: -2.0
TimeCost: 2
TotalMoney: 2600.0
TotalHealth: 77.5
TotalTime: 27


Process finished with exit code 0
```

شرح الكود:

- **Transport – Bus – Taxi – Walk**

قسم عمر الحكيم

الكلاس Transport هو كلاس مجرد abstract class يحوي الواصفات المشتركة بين وسائل النقل المتاحة وهي :



مقدار تغير الكلفة في المال MoneyCost

مقدار تغير الكلفة في الصحة HealthCost

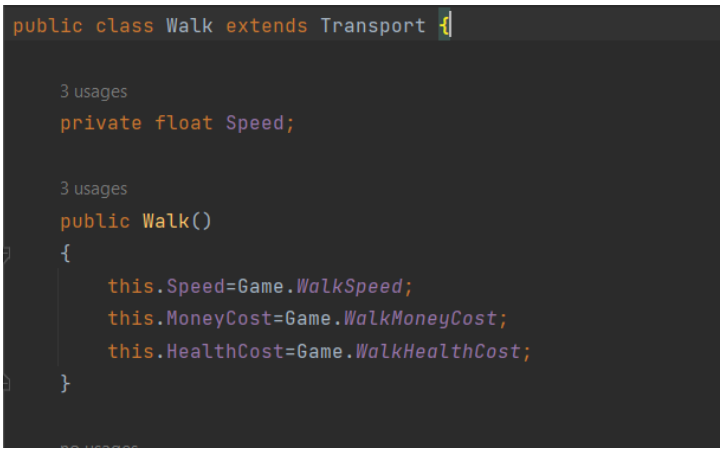
الكلاسات الثلاثة ترث من الكلاس Transport

وإن احتاجت لواصفة غير متشركة نقوم بإضافتها

في كل كلاس على حدى

وبداخل الConstructer يتم إسناد القيم الافتراضية

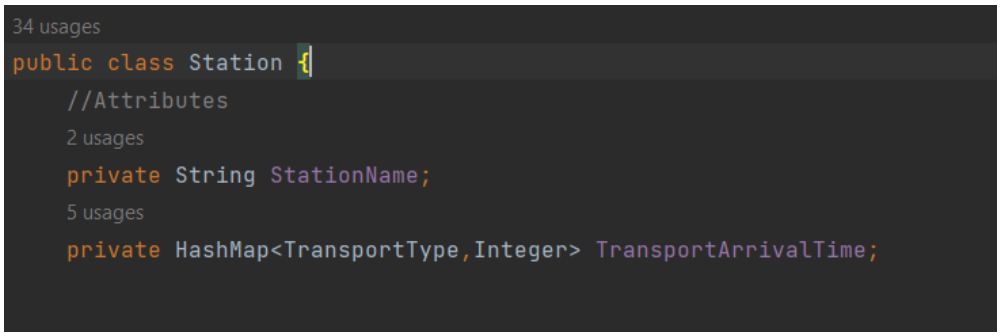
الموجود كواصفات ستاتيكة في كلاس Game



- **Station – Path - Person**

قسم محمد الحلاق

الكلاس **Station** يعبر عن المحطة ويحوي الواصفات :اسم المحطة ووقت الانتظار لكل وسيلة نقل



TranportType هو Enum كي لتقليل حجم التخزين بدلاً من تخزين ال Transport Object كاملاً

```java
public enum TransportType {

    Walk,

    Taxi,

    Bus
}
```

تابع GetArrivalTime يقوم بأخذ نوع الوسيلة كبارميتر ويعيد قيمة الانتظار لها في هذه المحطة

```java
1 usage
public int GetArriavalTime(TransportType Transport)
{
    if(TransportArrivalTime.get(Transport)==null)return 0;
    return TransportArrivalTime.get(Transport);
}
```

الكلاس **Path** يعبر عن الطريق الواصل بين محطتين

ويحوي الواصفات التالية :

محطة البداية – محطة النهاية – المسافة بين المحاطتين

هل وسائل النقل متاحة (تاكسي –باصات) –

أسماء خطوط الباصات – سرعة الباص – سرعة التاكسي

```java
public class Path {
    3 usages
    private Station StartStation;
    3 usages
    private Station EndStation;
    7 usages
    private float Distance;
    2 usages
    private boolean TransportEnabled;
    3 usages
    private List<String> BusNames;
    3 usages
    private float TaxiSpeed;
    3 usages
    private float BusSpeed;
```

ويحوي 3 توابع لحساب القيمة الجديدة لكل من المال والوقت والصحة بعد سلوك هذا الطريق

```
public int CalcTimeBetweenStations(Transport Transport){
    if(Transport instanceof Taxi)return Math.round((Distance/TaxiSpeed)*60);
    else if(Transport instanceof Bus)return Math.round((Distance/BusSpeed)*60);
    else return Math.round((Distance/new Walk().GetSpeed())*60);
}

1 usage
public float CalcNewMoneyCost(Transport Transport){
    if(Transport instanceof Taxi)return Transport.GetMoneyCost()*this.Distance;
    return Transport.GetMoneyCost();
}

1 usage
public float CalcNewHealthCost(Transport Transport) { return this.Distance*Transport.GetHealthCost(); }
```

الكلاس **Person** يحوي واصفتين قيمة المال الحالية الموجودة في الجيب وقيمة الصحة الحالية

```
public class Person {
    3 usages
    private float Health;
    3 usages
    private float Money;

    3 usages
    public Person()
    {
        Health=Game.PersonHealth;
        Money=Game.PersonMoney;
    }
```

- **Game**

قسم بشر جمعة

كلاس Game هو كلاس يحوي جميع القيم الافتراضية وجميع المدخلات كقيم static يمكن استخدامها في أي كلاس آخر ، تم إدخال المدخلات بحسب السيناريرو المرفق في أول التقرير وبالتالي جميع القيم يمكن التعديل عليها بسهولة

الواصفات:

```
39 usages
public static List<Station> Stations;
18 usages
public static List<Path> Paths;
14 usages
public static List<String> BusesNames;
```

المحطات الموجودة ضمن السيناريو

الطرق الموجودة ضمن السيناريو

أسماء خطوط الباصات

```
5 usages
public static Station FinishStation;
3 usages
public static Station FirstStation;
```

محطة النهاية

محطة البداية

```
public static float BusMoneyCost;
3 usages
public static float BusHealthCost;

5 usages
public static float TaxiMoneyCost;
3 usages
public static float TaxiHealthCost

5 usages
public static float WalkSpeed;
5 usages
public static float WalkMoneyCost;
3 usages
public static float WalkHealthCost
```

كلفة الباص المالية

كلفة الباص الصحية في كل 1 كم

كلفة التكسي المالية في كل 1 كم

كلفة التكسي الصحية في كل 1 كم

سرعة الشخص في المشي

كلفة المشي المالية

كلفة المشي الصحية في كل 1 كم

```
3 usages
public static float PersonHealth;
6 usages
public static float PersonMoney;
```

الطاقة الابتدائية للشخص

قيمة المال الابتدائية للشخص

```
12 usages
public static int cost;
2 usages
public static float WorstTime;
```

رقم الطلب (1 الكلفة بحسب الزمن

2 الكلفة بحسب المال

3 الكلفة بحسب الصحة

4 الكلفة بحسب جميع القيم)

Worst Time هي قيمة يتم حسابها مرة واحدة عند بدء تشغيل اللعبة وهي قيمة تعبر عن أسوء وقت ممكن تحصيله للوصول إلى المنزل بالسيناريو المدخل

وهي تفيد في ضبط مقام الوقت أثناء تحويل الكلف الثلاث إلى نسب مئوية في الطلب الرابع والتي سيمر الحديث عنها ضمن توابع الكلاس State

التوابع :

```
1 usage
public static void InitilizeGame()
{
    Stations=new ArrayList<>();
    Paths=new ArrayList<>();
    BusesNames=new ArrayList<>();

    FinishStation= new Station();

    DeclareBusesNames();
    DeclarePerson();
    DeclareTransports();
    DeclareStations();
    DeclarePaths();

    CalcWorstTime();
}
```

تابع تهيئة اللعبة يتم استدعاءه أول

تابع الMain لتهيئة المدخلات

تابع تعريف أسماء الباصات

تابع تعريف قيم تخص الشخص

تابع تعريف وسائل النقل

تابع تعريف المحطات

تابع تعريف الطرق

حساب أسوء وقت (يفيد في الطلب 4)

توابع تحديد رقم الطلب

```
no usages
static public void SetCostTime() { cost=1; }
no usages
static public void SetCostMoney() { cost=2; }
no usages
static public void SetCostHealth() { cost=3; }
1 usage
static public void SetCostAll() { cost=4; }
```

- **State**

قسم براء دركزلي

كلاس State يعبر عن العقدة التي سيتيم إنشاء الغراف على أساسها والتي قيمها ستتغير باختلاف العقدة ضمن الGraph

الواصفات:

```
public class State {
    22 usages
    private Person Person;
    12 usages
    private int Time;
    18 usages
    private Pair<TransportType,String>Transport;
    9 usages
    private State FatherState;
    12 usages
    private Station Station;
```

الشخص (قيمة الصحة ، قيمة المال)

الوقت المستغرق حتى الآن

وسيلة النقل التي وصلنا بها إلى هذه العقدة

مؤشر للعقدة الأب

اسم المحطة التي نحنا بها الآن

أهم التوابع:

```
//CheckMove
1 usage
public HashMap<Path,HashMap<TransportType,Boolean>> CheckMovesTransports(){
    HashMap<Path,HashMap<TransportType,Boolean>> CheckMoves=new HashMap<>();
    for(Path Path : Game.Paths)
    {
        HashMap<TransportType,Boolean>Transports=new HashMap<>();
        if(Path.GetStartStation().GetStationName()!=this.Station.GetStationName())continue;
        Transports.put(TransportType.Walk, v: true);
        if(Path.GetTransportEnabled()){
            Transports.put(TransportType.Taxi, v: true);
            if(Path.GetBusNames().isEmpty())Transports.put(TransportType.Bus, v: false);
            else Transports.put(TransportType.Bus, v: true);
        }
        else {
            Transports.put(TransportType.Taxi, v: false);
            Transports.put(TransportType.Bus, v: false);
        }
        CheckMoves.put(Path,Transports);
    }
    return CheckMoves;
}
```

Check Moves

يعيد Nested Hash Map تعبر عن كل وسيلة نقل هل هي متاحة أم لا في كل طريق يمكن سلوكه من المحطة التي نحن بها الآن

Move

```
//Move
2 usages
public void Move(Path Path){...}
```

يقوم بالتحرك إلى الPath الذي تم تمريره كParameter

ويتم من خلاله:

1- إضافة وقت الانتظار إلى الوقت الجديد في حال كانت وسيلة النقل مختلفة عن الوسيلة السابقة

```
//Add Wait Time
if(this.FatherState.Transport.getKey()!=this.Transport.getKey() ||
        (
            this.FatherState.Transport.getKey()==TransportType.Bus &&
            this.Transport.getKey()==TransportType.Bus &&
            this.FatherState.Transport.getValue()!=this.Transport.getValue()
        )
    )
{
    Time+=Station.GetArriavalTime(this.Transport.getKey());
}
```

2- تغيير قيمة المال الجديدة للشخص (في حال كان الشخص راكباً لنفس خط السرفيس السابق لن يتم حساب كلفة جديدة)

```
//Change Person Money
if(this.Transport.getKey()==TransportType.Bus && this.FatherState.Transport.getKey()==this.Transport.getKey())
{
    if(this.FatherState.Transport.getValue()==this.Transport.getValue())SameBus=true;
}
if(!SameBus)
{
    this.Person.SetMoney(this.Person.GetMoney()-Path.CalcNewMoneyCost(Transport));
}
```

3- حساب قيمة الوقت الجديدة

4- حساب قيمة صحة الشخص الجديدة

5- تغيير المحطة الحالية إلى محطة النهاية لدى الطريق المسلوك

```
//Change Time
Time+=Path.CalcTimeBetweenStations(Transport);

//Change Person Health
this.Person.SetHealth(this.Person.GetHealth()-Path.CalcNewHealthCost(Transport));

//Change Station
this.Station=Path.GetEndStation();
```

GetNextStates

ملخص عمل التابع

يعيد List of States لكل الحالات الممكن سلوكها باعتبار محطة البداية هي المحطة التي نقف بها حالياً ، وبداخل التابع يتم التحقق من شرط

إذا كانت الحالة الجديدة قيمة المال فيها أصغر أو يساوي 0

أو كانت قيمة الصحة فيها أصغر أو تساوي 0

فسيتم تجاهل هذه الحالة

Is Finish يعيد True إذا كان المحطة الحالية هي محطة النهاية

Is Equal يقارن بين Two States

CalcHuristic يقوم بحساب القيمة التقريبية لإضافتها مع الكفلة في خوارزمية ال*A

تم حساب قيمة الهيورستيك من خلال الخوارزمية الطماعة Greedy وذلك باعتبار الهدف هو أقصر مسافة

```
//CalcHeuristic
2 usages
private float CalcHeuristic(Station Station,float distance){
    if(Station.GetStationName()==Game.FinishStation.GetStationName())return distance;
    float MinDistance=Float.MAX_VALUE;
    Station NextStation=new Station();
    for(Path Path : Game.Paths)
    {
        if(Path.GetStartStation().GetStationName()!=Station.GetStationName())continue;

        if(Path.GetDistance()<MinDistance){
            MinDistance=Path.GetDistance();
            NextStation=Path.GetEndStation();
        }
    }
    return CalcHeuristic(NextStation, distance: MinDistance+distance);
}
```

تابع CalcCostAll

يقوم بحساب الكلفة كونها مجموع الكلف الثلاث بالسالب بعد تحويلها إلى نسب مئوية

باعتبار مقام نسبة المال هي قيمة المال الابتدائية

ومقام نسبة الصحة هي قيمة الصحة الابتدائية (100)

ولحساب قيمة مقام نسبة الزمن قمنا عند بدء اللعبة بحساب أسوء وقت يمكن من خلاله الوصول إلى المنزل (باستخدام خوارزمية A* وجعل الكلفة هي أطول وقت) (يتم حساب الخوارزمية مرة واحدة عند بدء اللعبة)

واعتباره هو معيار التأخر في هذا السيناريو

```
private float CalcCostIsAll()
{
    return -(Person.GetMoney()*100/Game.PersonMoney)
           -(Person.GetHealth()*100/Game.PersonHealth)
           -((100-Time)*100/Game.WorstTime);
```

- **Logic**

قسم أحمد البغدادي

تابع الLogic لحساب الخوارزميات المراد تطبيقها لحل المسألة

تم تكويد 3 خوارزميات A* – UCS – BFS

أهم الواصفات:

List بالحالات المزارة

Stack لتخزين مسار الحل

وطباعته

```
6 usages
private List<State>VisitedStates=new ArrayList<>();
6 usages
private Stack<State>Solution=new Stack<>();
4 usages
public LocalTime StartRunTime;
4 usages
public LocalTime StopRunTime;
```

بالإضافة إلى زمن التنفيذ ، يتم بدء حساب زمن التنفيذ بمجرد بدء الخوارزمية ويتم إنهاءه بمجرد الوصول إلى الحالة النهائية

كود خوارزمية A* :

```
public void A_Star(State State){
    StartRunTime=LocalTime.now();
    PriorityQueue<Pair<Float, State>> pq = new PriorityQueue<>(Comparator.comparing(Pair::getKey));
    pq.add(new Pair<>(State.GetCost()+State.GetHeuristic(),State));

    while(pq.size()!=0)
    {
        State v=new State();
        v=pq.remove().getValue();
        if(!isVisited(v))
        {
            VisitedStates.add(v);
            if(v.IsFinish())
            {
                FinishStateCost=v.GetCost();
                StopRunTime=LocalTime.now();
                GetSolution(v);
                Algorithm="A*";
                return;

            }
            for(State next : v.GetNextStates())
            {
                pq.add(new Pair<>(next.GetCost()+next.GetHeuristic(),next));
            }
        }

    }
}
```

النهاية