# MiniJava Documentation

## Introduction

- زبان MiniJava یک زبان شی گراست که برخی از قابلیت های زبان java را داراست.
- هر برنامه زبان MiniJava شامل یک فایل با فرمت jm. است
- گرامر زبان در این قسمت قابل مشاهده است.

## Class Declaration

- تعریف هر کلاس از دو بخش نام و بدنه تشکیل شده است. بدنه هر کلاس شامل تعریف فیلد ها و متد ها می باشد.
- هر کلاس می تواند حداکثر از یک کلاس ارث بری کند (با استفاده از کلید واژه extends ) به این معنی که می تواند از فیلد ها و متد های پدرش استفاده نماید.
- همچنین یک کلاس میتواند از یک یا چند interface را پیاده سازی کند (با استفاده از کلمه کلیدی implements).

```
class Pokemon{
    // Field Declaration
    // Method Declaration
}
```

## Import Class

- برای استفاده از کلاس های از پیش تعریف شده باید آنها را قبل از شروع کلاس ها import کنیم.

```
import Pokemon;
import Digimon;
```

## Interface Declaration

- تعریف آن همانند کلاس بود با این تفاوت که متد های آن به صورت abstract می باشد و از آنها برا ساخت ابجکت نمی توان استفاده کرد.
- عملیات انتساب فیلد های یک interface دقیقا پس از تعریف آنها صورت می گیرد.
- فیلد های یک اینترفیس به طور پیش فرض static, public و final(فقط یک بار مقداردهی می شوند) هستند
- کلاسی که اینترفیس را implements می کند، باید تمامی متدهای آن را پیاده سازی کند.
- اینترفیس نمی تواند از کلاسی ارث ببرد.

```
interface Ball {
    // Field Declaration
    // Interface Method Declaration
}
```

```
class PokeBall implements Ball {
    ...
}
```

## Inheritance

- هنگامی که یک کلاس از کلاس دیگر ارث می برد کلاس فرزند می تواند به خصوصیات و متد های کلاس پدر دسترسی داشته باشد مگر آنکه دسترسی آنها private باشد.
- متد های پدر در صورت نیاز میتوانند override شوند. ( با استفاده از کلید واژه @Override)
- قوانین override کردن:
  - لیست آرگومان های تابع باید دقیقا با لیست آرگومان های تابع override شده یکی باشد.
  - نوع مقدار بازگشتی تابع باید همانند نوع مقدار بازگشتی تابع override شده کلاس پدر باشد.
  - سطح دسترسی تابع نمی تواند محدودتر از سطح دسترسی تابع override شده باشد. برای مثال: اگر تابع کلاس پدر به صورت public تعریف شده باشد، در این صورت تابع کلاس فرزند نمی تواند private باشد.
  - تنها توابعی از کلاس پدر که توسط فرزند ارث بری شده اند می توانند override شوند.

```
// Base Class
class Pokemon {
    public int damage() {
    health = health - 5;
    return health;
    }
}
// Inherited class
class Pikachu extends Pokemon {
    @Override
    public int damage() {
    health = health - 2;
    return health;
    }
}
```

## Main Class Declaration

- هر برنامه شامل فقط یک کلاس main می باشد که به صورت تعریف می شود
- کلاس main تنها یک متد main دارد وخروجی آن void می باشد. همچنین این متد تنها متد static در برنامه می باشد.

```
class ClassName{
    public static void main(String[] a){
    //statement
    }
}
```

## Field Declaration

- هر کلاس دارای تعدادی خصیصه است که در زمان تعریف حتما نوع آن باید تعیین شود.
- خصیصه ها فقط درون کلاس قرار میگیرند و نباید خصیصه های تکراری درون یک کلاس تعریف شود.

```
class Pokemon {
   int health;
   final int maxMovesNum = 5;
   private int speed;
   boolean legendary;
   Pokeball ball;
   int[] powers = {1, 2, 3};
   Move[] moves = new Moves[maxMovesNum];
}
```

## Local Variable Declaration

- متغیر ها همانند خصیصه ها مقدار دهی می شوند با این تفاوت که متغیر ها داخل متد ها تعریف می شوند

```
public attack() {
    int hp;
}
```

## Method Declaration

- هر متد تعدادی ورودی و یک خروجی دارد، نوع ورودی ها و خروجی باید مشخص باشد
- در صورتی که تابع خروجی نداشته باشد از کلید واژه void استفاده میشود
- عبارت return حداکثر یک بار و در آخر تابع ظاهر می شود

```
public int damage(int amount) {
    health = health - amount;
    return health;
}

public static void attack(Pokemon target, int amount) {
    target.damage(amount);
}
```

## Abstract Method Declaration

- متد های interface بدنه ندارند
- همچنین در هنگام تعریف به صورت پیش فرض abstract و public)هستند و هیچگاه به صورت private نمیتوان آنها را تعریف کرد

```
interface Pokemon {
    void speedUp(int a);
}
```

# Statements

**If Else Statement**

```
if (health < 1) {
  isAlive = 0
}
```

**While Statement**

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

## Assignmets Statement

**Variable Assignment Statement**

- برای استفاده از ابجکت ها باید ابتدا آنها را instantiate کرد

```
health = 100;
powers = new int[20];
ball= new PokeBall();
amount = power[4];
```

**Array Assignment Statement**

```
power[1] = 5;
```

## This Keyword

کلید واژه this به کلاسیکه در آن قرار داریم اشاره می کند
راه های استفاده از this در Minijava به شرح زیر می باشد:

- ای از کلاس فعلی instace برای برگرداندن

```
class Test
{
    //Method that returns current class instance
    Test get()
    {
        return this;
    }
}
```

- دسترسی به متد کلاس

```
class Test {
    void display()
    {
        // calling function show()
        this.show();
    }
    void show() {
        ...
    }
}
```

## Data Types

زبان MiniJava که در اختیار شماست شامل دو نوع داده ای پایه Boolean و int می باشد. علاوه برآن هر متغیر میتواند از جنس یکی از کلاس هایی باشد که در برنامه تعریف شده است. در این زبان، یک نوع آرایه نیز تعریف شده است. این آرایه، یک بعدی و می تواند از هر نوعی باشد.

| نوع | توضیحات | مقدار پیش فرض |
|---|---|---|
| int | یک نوع عدد دهدهی که با یک رقم 1 تا 9 شروع و می تواند با ارقام 0 تا 9 ادامه یابد همچنین رقم 0 نیز می تواند باشد | 0 |

| نوع | توضیحات | مقدار پیش فرض |
|---|---|---|
| boolaen | نوع داده ای شامل true و false می باشد | fasle |
| array[] | آرایه ای یک بعدی از هر گونه | - |
| Class | داده ای از نوع کلاسهای تعریف شده در برنامه | null |

- در صورتی که متغیر از جنس یک کلاس یا آرایه باشد، مقدار اولیه ندارد و در صورت استفاده، باید خطای مناسب به کاربر داده شود.
- اندازه یک آرایه نمی تواند صفر یا عددی منفی باشد.

- کلاس هاست) انجام شود scope که scope غیر از بیرونی ترین) scope تعریف متغیر ها می تواند در هر کجای

## Predefined Methods

در زبان Minijava توابع و فیلدهای از پیش تعریف شده ای وجود دارند که عبارت اند از:

**println Method**

این تابع به طور ضمنی تعریف شده است و می تواند یک مقدار intو boolean را دریافت کند وآن را بر روی کنسول چاپ کند

```
System.out.println(amount);
```

**length field**

این فیلد تنها برای آرایه ها تعریف می شود و طول یک آرایه را باز می گرداند.

```
aray = new int[66];
print(array.length);
```

## Operators

**Arithmetic operators**

در صورتی که A=10 و B=2 باشد، عملگر های زبان MiniJava را به صورت زی تعریف میکنیم

| Operator | Description | Example |
|---|---|---|

| Operator | Description | Example |
|---|---|---|
| +(Adiituion) | Addss values on eithr side of theoperator | A + B will give 12 |
| -(Sibtraction) | Subtracts right-hand operand from left-hand operand | A - B will give 8 |
| *(Multiplication) | Multiplies values on eithr side of the operator | A * B will give 20 |
| ** (Exponentiation) | raise the number on the left to the power of the exponent of the right | A ** B will give 100 |

**Assignment Operator**

در زبان MiniJava تنها یک دستور انتساب تعریف شده است

| Operator | Description | Example |
|---|---|---|
| = | Simple assingment operator. Assigns values from right side operands to left side operand | C= A + B will assign value of A + B into C |

**Logical Operators**

| Operator | Description |
|---|---|
| && | returns the boolean value TRUE if both operands are TRUE and returns FALSE otherwise. |

**Relational Operators**

| Operator | Description | Example |
|---|---|---|
| <(Less Than) | Checks if the value of the left opernad is less than the value of right operand, if yes then the condition becoms true. | (A < B) is true |