**Mini java**

* 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.

**تعریف کلاس**

* تعریف هر کلاس از دو بخش نام و بدنه تشکیل شده است. بدنه هر کلاس شامل تعریف [فیلد](https://github.com/NajmeHabibi/CompilerDesignProjectDescription/tree/master/Documentation" \l "field-declaration) ها و [متد](https://github.com/NajmeHabibi/CompilerDesignProjectDescription/tree/master/Documentation" \l "method-declaration) ها می باشد.
* هر کلاس می تواند حداکثر از یک کلاس ارث بری کند (با استفاده از کلید واژهinherits) ) 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).
* کلاس Object به صورت پیش فرض پدر تمامی کلاس ها می باشد.

class Pokemon{

// Field Declaration

// Method Declaration

}

**تعریف اینترفیس**

* 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 ) finalفقط 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 {

...

}

**ارث بری**

* 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 number damage() {

health = health - 5;

ret health;

}

}

// Inherited class

class Pikachu inherits Pokemon {

@Override

public number damage() {

health = health - 2;

ret health;

}

}

**کلاس main**

* 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

}

}

**تعریف متغیر**

* 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.
* خصیصه ها به طور پیش فرض public می باشند.
* خصیصه ها می توانند انواع boolean، number داشته باشند.

class Pokemon {

number health;

final number maxMovesNum = 5;

private number speed;

boolean legendary;

Pokeball ball;

number[] powers = {1, 2, 3};

Move[] moves = new Moves[maxMovesNum];

}

**تعریف متغیر محلی (local variable)**

* متغیر ها همانند خصیصه ها مقدار دهی می شوند با این تفاوت که متغیر ها داخل متد ها و nested stament ها تعریف می شوند.

public attack() {

number hp;

}

**تعریف متد**

* 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 می شود.
* عبارت ret حداکثر یک بار و در آخر تابع ظاهر می شود.
* در صورت عدم تعریف سطح دسترسی متد ها به صورت پیش فرض پابلیک می باشند.

public number damage(number amount) {

health = health - amount;

ret health;

}

public static void attack(Pokemon target, number amount) {

target.damage(amount);

}

**تعريف متد abstract**

* متد های interface بدنه ندارند.
* همچنین در هنگام تعریف به صورت پیش فرض abstract و public هستند و هیچگاه به صورت private نمی توان آنها را تعریف کرد.

interface Pokemon {

void speedUp(number a);

}

**Statements**

* **If Else Statement**

if (health < 1) {

isAlive = 0

}

* **While Statement**

number i = 0;

while (i < 5) {

print(i);

i++;

}

**انتساب**

* **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;

**كليد واژه this**

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

* برای برگرداندن instace ای از کلاس فعلی

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() {

...

}

}

**انواع داده**

زبان MiniJava که در اختیار شماست شامل دو نوع داده ای پایه Boolean و number 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** |
| --- | --- | --- |
| number | 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 |
| 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.
* تعریف متغیر ها می تواند در هر کجای scope (غیر از بیرونی ترین scope که scope کلاس هاست) انجام شود.

**نكات**

* در زبان مینی جاوا به طور پیش فرض کلاس های String و Object وجود دارند.
* در زبان Minijava توابع و فیلدهای از پیش تعریف شده ای وجود دارند که عبارت اند از:

**print Method**

این تابع به طور ضمنی تعریف شده است و می تواند یک مقدار دعئذثفnumber، boolean, string را دریافت کند وآن را بر روی کنسول چاپ کند.

print(amount);

**length field**

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

aray = new number[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** |
| --- | --- | --- |
| +(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. |
| <> | not |

**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 |

**Comments**

در زبان مینی جاوا دو نوع کامنت گذاری وجودا دارد.

* starts with /\*, ends with \*/, and may be nested
* begins with // and goes to the end of the line