**BSCS-504*Compiler Construction***

***LANGUAGE SPECIFICATIONS***

**Operators:**

|  |  |
| --- | --- |
| **Symbol** | **Description** |
| + | Addition, Concatenation |
| - | Subtraction |
| / | Division |
| \* | Multiplication |
| % | Remainder |
| = | Assignment |
| == | Comparison |
| ++ | Increment |
| -- | Decrement |
| // | Integer Division |
| && | Logical OR |
| || | Logical AND |
| # | Single line comment |

**Keywords:**

|  |  |  |
| --- | --- | --- |
| void | abstract class | yield |
| main | function | resume |
| int | For | interrupt |
| float | until | if |
| word | do | else |
| boolean | In | elif |
| char | public | when |
| class | private |  |

**Punctuators:**

|  |  |  |
| --- | --- | --- |
| ( | [ | ; |
| ) | ] | : |
| { | .(dot) |  |
| } | , |  |

**Expressions:**

|  |  |
| --- | --- |
| **Expression** | **Description** |
| x \* y | Multiplication |
| x / y | Division |
| x % y | Remainder |
| x + y | Addition, concatenation |
| x - y | Subtraction |
| x = y | Assignment |
| x == y | Comparison |
| x && y | boolean logical AND |
| x || y | boolean logical OR |
| x != y | Not equal |

**Data Types:**

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Description** | **Equivalent in c#** |
| int | Stores numbers | int |
| float | Stores floating point | float |
| word | Stores strings | string |
| char | Stores character | Char |
| boolean | Store boolean value | bool |

**Loops:**

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Description** | **Equivalent in Java** |
| for | for(i in 1..10)  {  //your code here  } | For(int i=1 ;i<=10;i++){  // code  } |
| for | for(i in 1..10 step 2)  {  //your code here  } | For(int i=1 ;i<=10;i+=2){  // code  } |
| until | until(boolean exp)  {  //code to be executed  } | while(boolean exp){  //code  } |
| do  until | do{  //code  }until(bool exp) | do{  // code  }while(boolean exp) |

**Branching Statements:**

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Description** | **Equivalent in Java** |
| if | If(boolean exp){  // code  } | Same as mentioned |
| else | else{  // code  } | Same as mentioned |
| elif | elif(boolean expression){  // code  } | Else if(boolean expre){  // code  } |
| when | when(value){  1->{ //code}  2->{//code}  3->{//code}  else->{//code}  } | switch(value){  case 1:{//code break;}  case 2:{//code break;}  case 3:{//code break;}  default:{//code}  } |

**Return and Jumps:**

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Description** | **Equivalent in Java** |
| yield | yield value; | return value; |
| yield | yield; | return; |
| interrupt | Interrupt(Terminate the nearest enclosing loop) | break; |
| resume | resume(Proceeds to the next step of the nearest enclosing loop) | continue; |

**Statements:**

|  |  |
| --- | --- |
| Incremental statement | int a = 1;  a++;  //value of a is 2 now |
| Comment statement | # is used for comments  Scope of a comment is only the current line |
| Terminator | ; is used for terminating a statement |

**Functions:**

|  |  |
| --- | --- |
| function | Function <name>(): <Yield type>{  //function body} |

**Object Oriented Feature:**

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| class | class <classname>{  // class body  } |
| abstract class | abstract class <classname>{  //class body  } |
| inheritance | classA extends classB{  // body  } |
| constructor | classA{  classA(){  //constructor body  }} |
| public | public (Access Modifier) visible outside class. |
| private | private (Access Modifier) visible only inside class |
| Object Creation | classA object=classA(); |