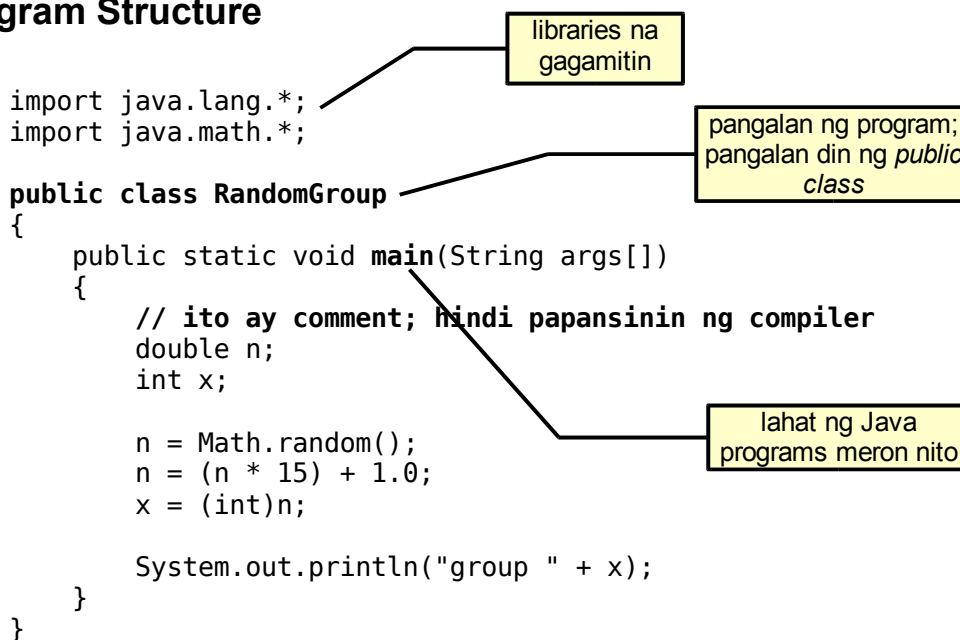


Keywords

<code>import</code>	<code>for</code>	<code>static</code>
<code>main</code>	<code>do</code>	<code>volatile</code>
	<code>while</code>	<code>final</code>
<code>void</code>	<code>continue</code>	
		<code>class</code>
<code>char</code>	<code>if</code>	<code>extends</code>
<code>byte</code>	<code>else</code>	
		<code>interface</code>
<code>int</code>	<code>switch</code>	<code>implements</code>
<code>long</code>	<code>case</code>	
<code>short</code>	<code>break</code>	<code>return</code>
		<code>goto</code>
<code>float</code>	<code>public</code>	
<code>double</code>	<code>private</code>	<code>try</code>
	<code>protected</code>	<code>catch</code>
		<code>finally</code>

- hindi muna natin gagamitin ang *italicized* keywords

Program Structure



Tandaan:

- Ang pangalan ng **public class** ay pareho sa pangalan ng file name. Kahit ang capitalization dapat pareho. Halimbawa: kung **HelloWorld** ang pangalan ng public class, **HelloWorld.java** dapat ang file name.
- Ang extension ng source code file ay **java**. Hindi tatakbo ang program kung iba.
- Kahit ilang **class** ang pwede sa loob ng isang source code file; subalit, isang **public class** lang ang pwede.
- Para madalian tayong lahat, iisang **main** lang ang gawin nyo at sa public class nyo ito ilagay.

Data Types (aka Primitives, Built-in Types)

```
int a;      // integer (whole numbers)
double b;   // rational (may fractional part)
char c;     // isang character
boolean d;  // true or false
```

Tandaan:

- Marami pang ibang data types. Itong apat lang ang gagamitin natin.
- Kailangan napapaligiran ng single-quotes ang characters (hal: 'O').
- Pwedeng mag-convert ang **int** papuntang **double**. Hindi pwedeng mag-convert ang **double** papuntang **int**.
- Para madaling mabasa, gawing isang variable bawat lina.

Operators

Equality

< less than
> greater than
<= less than or equal to
>= greater than or equal to
!= not equal to
== equal to

Logic

&& AND
|| OR
! NOT
& *bitwise AND*
| *bitwise OR*

Math

+ addition
- subtraction
++ increment
-- decrement

* multiplication
/ division
% modulo

- hindi muna natin gagamitin ang *italicized* operators

Loops

```
while (expression)
{
    // do something
    // paulit-ulit habang true ang expression
}

for (initial; expression; increment/decrement)
{
    // do something
    // paulit-ulit habang true ang expression
}
```

Halimbawa: **while** loop at ang katumbas na **for** loop

```
int c = 0;
while (c < 100)
{
    System.out.println("quack");
    c = c + 1;
}

int c;
for (c = 0; c < 100; c = c + 1)
{
    System.out.println("quack");
}
```

Tandaan:

- Hindi kailangan ng *increment/decrement* sa **while** loop, pwedeng wala. Siguraduhin nyo lang na magbabago ang *expression*; infinite loop ang mangyayari kung hindi.
- Gamitin ang **for** loop kung alam nyo kung ilang beses uulit ang loop.
- Gamitin ang **for** loop kung may binibilang ka habang umuulit ang loop.
- Gamitin ang **while** loop kung di mo alam kung ilang beses uulit ang loop.
- Gamitin ang **while** loop kung may hinihintay kayo na magbabago sa *expression*.

Halimbawa: loop na may hinihintay

```
boolean puzzleSolved = false;

while (!puzzleSolved)
{
    // do something
    // meron ditong code na magbabago sa puzzleSolved; parang ganito
    //   if(something)
    //   {
    //       puzzleSolved = true;
    //   }
}
```

Execution Control (if ... else)

```
if (expression)
{
    // do something
    // gagawin kapag true ang expression
}
else
{
    // do something
    // gagawin kapag false ang expression
    // optional ang else; kung walang else, walang gagawin
}
```

Halimbawa:

```

if(score < 100)
{
    System.out.println("low score");
    count = count + 5;
}
else if ((score >= 100) && (score < 200))
{
    System.out.println("medium score");
    count = count + 10;
}
else
{
    System.out.println("high score");
    count = count + 15;
}

```

halimbawa ng magkadugtong na **else** at isa pang **if**

halimbawa ng komplikadong **expression**

Tandaan:

- Hindi kailangan ang **else**; pwedeng walang **else**.
- Kung merong **else**, dapat may kapares na **if**.
- Pwedeng complicated ang **expression**. Lagyan ng *parenthesis* para malinaw.
- Para malinaw, kahit isa lang ang ginagawa ng **if** o **else** lagyan ng *curly braces*.

Execution Control (switch ... case)

```

switch(variable)
{
    case value:
        // do something
        // gawin ito kapag value ang laman ng variable
        break;
    case ibang_value:
        // do something
        // gawin ito kapag ibang_value ang laman ng variable
        break;
    default:
        // do stuff
        // pupunta lang dito kapag walang case na pinasukan
        break;
}

```

Notes:

- Iisang variable lang ang kayang tingnan ng **switch**.
- **Integers** at **characters** lang ang kayang tingnan ng switch. Kung may kailangan kayong tingnan na **double** o **boolean**, **if ... else** ang dapat gamitin.
- Equality (==) lang ang kayang tingnan ng switch. Hindi nito kaya ang inequality (<, >, <=, >=) at iba pa (!=).
- Constant lang ang kayang ikumpara ng switch. Hindi pwedeng variable ang ilagay mo sa **case**.

- Kapag walang **break** sa loob ng **case**, diretcho ito sa susunod na **case**; hindi na tinitingnan kung ano ang **case**.
- Parating maglagay ng **default** case. Makakatulong ito sa debugging ng code; hinuhuli ng **default** case ang lahat ng hindi pumasok ng ibang **case**.

Halimbawa:

```

char grade;
switch(grade)
{
    case 'A':
        Console.println("Outstanding");
        break;
    case 'B':
        Console.println("Excellent");
        break;
    case 'C':
        Console.println("Satisfactory");
        break;
    case 'D':
        Console.println("Passing");
        break;
    case 'F':
        Console.println("Failed");
        break;
    default:
        Console.println("Error. There is no such grade.");
        break;
}
    
```