পাঠ ৩: ডাটা টাইপস এবং অপারেটর

* ভেরিয়েবল
* প্রিমিটিভ ডাটাটাইপ - ইন্টিজার, লং, ডাবল, ইন্টিজার, ফ্লোট এবং কার ইত্যাদি।
* র‍্যাপার ক্লাস
* লিটারেল
* বিভিন্ন রকম অপারেটর

**ভ্যারিয়বল**

ভ্যারিয়বল হচ্ছে একটি নাম যা কম্পিউটারের একটি মেমোরি লোকেশান কে নির্দেশ করে। উদাহরণ-

int cadence = 0;

একটি ভ্যারিয়বল ডিক্লারেশন এর জন্যে একটি ডাটাটাইপ দরকার হয়, অর্থাৎ ভ্যারিয়বল টি কি ধরণের ডাটা হোল্ড করবে তা বলে দিতে হবে। উপরের উদাহরণটিতে আমরা একটি ভ্যারিয়বল ডিক্লার করেছি যার নাম cadence এবং এটি ইন্টিজার টাইপ ডাটা হোল্ড করে।

যেহেতু জাভা একটি স্ট্যাটিক্যালি টাইপড ল্যাংগুয়েজ সুতরাং ভ্যারিয়বল ডিক্লারেশন এর সময় ডাটাটাইপ উ্ল্লেখ করা অত্যাবশ্যক।

জাভাতে আমরা চার ধরণের ভেরিয়েবল নিয়ে কাজ করে থাকি -

1. Instance Variables (Non-static fields)
2. Class Variables (Static Fields)
3. Local variables
4. Parameters variables

জাভাতে ভ্যারিয়বল এবং ফিল্ড দুই শব্দই ব্যবহার করা হয়, তবে এর কিছু টেকনিকাল পার্থক্য আছে। সেগুলো নিয়েই আলোচনা করা হবে –

আমরা আবার একটি উদাহরণ দেখি –

public class Bicycle {

static int numGears = 6;

​

int cadence = 0;

int speed = 0;

int gear = 1;

​

public Bicycle() {

}

​

void changeCadence(int newValue) {

cadence = newValue;

}

​

void changeGear(int newValue) {

gear = newValue;

}

​

void speedUp(int increment) {

speed = speed + increment;

}

​

void applyBrakes(int decrement) {

speed = speed - decrement;

}

​

void printStates() {

System.out.println("cadence:" +

cadence + " speed:" +

speed + " gear:" + gear);

}

}

আমরা জানি যে একটি ক্লাস থেকে আমরা অনেকগুলো অবজেক্ট তৈরি করতে পারি। এবং প্রত্যেক অবজেক্ট-ই আলাদা আলাদা। যেমন -

Bicycle bike1 = new Bicycle();

Bicycle bike2 = new Bicycle();

এখানে bike1 এবং bike2 দুটি সম্পূর্ণ আলাদা দুইটি অবজেক্ট।

এখন bike1 এবং bike2 তে কিছু ভ্যারিয়বল গুলোও আলাদা। অর্থাৎ আমরা যতগুলো অজেক্ট তৈরি করবো ঠিক ততোগুলো আলাদা ভ্যারিয়বল থাকবে মেমোরিতে। এক্ষেত্রে মেমোরিতে ২টা cadence থাকবে, ২টা gear থাকবে এবং ২ speed ধাকবে।

এই ভ্যারিয়বল গুলোকে Instance Variables বা Non-static fields বলা হয়। এই ভ্যারিয়বল গুলো আগে static কিওয়ার্ডটি থাকে না।

static int numGears = 6;

উপরের উদাহরণটিতে **numGears** নামে একটি ভ্যারিয়বল আছে, এটির আগে একটি **static** কিওয়ার্ডটি আছে। এ ধরণের ভ্যারিয়বল কে Class Variables বা Static Fields বলা হয়। static কিওয়ার্ডটি কম্পাইলারকে বলে যে numGears নামে একটি মাত্র ভ্যারিয়বল থাকবে মেমোরিতে, অবজেক্ট এর সংখ্যা যতই হোক।

লোকাল ভ্যারিয়বল হলো সেসব ভ্যারিয়বল যে গুলো কোন মেথডের মাঝে ডিক্লার করা হয়। একটি লোকাল ভ্যারিয়বল শুধু মাত্র সেই মেথডের ভেতর থেকেই একসেস করা যাবে।

আর Parameters variables হলো সেই ভ্যারিয়বল গুলো যেগুলো মেথড কল করার সময় পাস করা হয়। এ গুলোও শুধুমাত্র মেথডের ভেতর থেকেই একসেস করা যায়।

আমরা Instance Variables এবং Class Variables গুলোকে ফিল্ড বলি।

এখানে কিছু ভ্যারিয়বল ডিক্লারাশেনের উদাহরণ দেওয়া হলো -

byte myByte;

short myShort;

char myChar;

int myInt;

long myLong;

float myFloat;

double myDouble;

boolean myBool;

শুরুতে আগে টাইপ লিখতে হবে, তারপর একটি নাম, তারপর সেমিকোলন দিয়ে শেষ করতে হবে। তবে আমরা চাইলে ভ্যারিয়বল কে ইনিশিয়ালাইজেশান করতে পারি। যেমন -

int cadence = 0;

অর্থাৎ শুরুতে আমরা cadence এর ভ্যালু 0 এসাইন করালাম।

এরপর যদি আমরা কোন ভ্যারিয়বলে ভ্যালু এসাইন করতে চাই তাহলে -

myByte = 127;

myFloat = 199.99;

জাভা ভ্যারিয়বল লেখার কিছু নিয়ম কানুন আছে-

1. ভ্যারিয়বল গুলো কেইস সেনসিটিভ। অর্থাৎ money, Money,MONEY তিনটি আলাদা।
2. ভ্যারিয়বল অবশ্যই যেকোন একটি লেটার দিয়ে শুরু করতে হবে। তবে $ অথবা \_ দিয়েও শুরু করা যায়।
3. ভ্যারিয়বল এর মাঝে নাম্বার কিংবা \_ থাকতে পারে।
4. ভ্যারিয়বল জাভার কোন reserved কিওয়ার্ড হতে পারবে না।

**ডাটা টাইপ**

জাভা তে আট ধরণের প্রিমিটিভ ডাটা টাইপ আছে ।

|  |  |
| --- | --- |
| Data type | Description |
| byte | 8 bit signed value, values from -128 to 127 |
| short | 16 bit signed value, values from -32.768 to 32.767 |
| char | 16 bit Unicode character |
| int | 32 bit signed value, values from -2.147.483.648 to 2.147.483.647 |
| long | 64 bit signed value, values from -9.223.372.036.854.775.808 to 9.223.372.036.854.775.808 |
| float | 32 bit floating point value |
| double | 64 bit floating point value |
| boolean | true & false |

এগুলো প্রমিটিভ ডাটা টাইপ, এর মানে হচ্ছে এগুলো অবজেক্ট নয়। এরা মেমোরিতে সরাসরি ভ্যালু রাখে।

**র‍্যাপার ক্লাস**

তবে জাভাতে কিছু ডাটা টাইপ আছে যেগুলো অবজেক্ট।

|  |  |
| --- | --- |
| Data type | Description |
| Byte | 8 bit signed value, values from -128 to 127 |
| Short | 16 bit signed value, values from -32.768 to 32.767 |
| Character | 16 bit Unicode character |
| Integer | 32 bit signed value, values from -2.147.483.648 to 2.147.483.647 |
| Long | 64 bit signed value, values from -9.223.372.036.854.775.808 to 9.223.372.036.854.775.808 |
| Float | 32 bit floating point value |
| Double | 64 bit floating point value |

এগুলোকে প্রিমিটিভ টাইপ এর র‍্যাপার ক্লাস বলা হয়। লক্ষ্য করুণ, এগুলোর সবগুলোর নাম ক্যাপিটাল অক্ষর দিয়ে শুরু হয়েছে।

তবে আমরা চাইলে অবজেক্ট ডাটাটাইপ এবং প্রিমটিভ ডাটাটাইপ একে অপরের পরিপূরক হিসাবে ব্যবহার করতে পারি।

Integer a;

int b = 9;

a = b;

তবে প্রিমিটিভ ডাটা টাইপ গুলোর ডিফল্ট ভ্যালু থাকে। অর্থাৎ আমরা যদি ভ্যালু এসাইন না করি, তাহলে এদের মধ্যে বাইডিফল্ট ভ্যালু থাকে। যেমন -

|  |  |
| --- | --- |
| Data Type | Default Value (for fields) |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| boolean | false |

**লিটারেল-**

প্রোগ্রামিং ল্যাংগুয়েজে কিছু মজার মজার বিল্ট-ইন সুবিধা থাকে, তার মধ্যে লিটারেল একটি। আমরা জানি যে একটা ভ্যারিয়বল ডিক্লারেশান এর জন্য প্রথমে টাইপ লিখতে হয়, তারপর একটা নাম দিতে হয়, তারপর একে ইনিশিয়ালাইজেশান করতে হয়। ভেরিয়েবলটি যদি অবজেক্ট হয়, তাহলে ইনটেনশিয়েশান করতে হয়।

উদাহরণ-

List list = new ArrayList();

​

or

​

Int x = 5;

উপরের দুটি উদাহরণের মাঝে একটিতে আমরা new কিওয়ার্ড ব্যবহার করে নতুন অবজেক্ট তৈরি করেছি। কিন্তু পরের উদাহরণটিতে সেটি করতে হয় নি। আমরা সরাসরি একটি ভ্যালু এসাইন করেছি। এখানে 5 একটি ভ্যালু। এখানে 5 হচ্ছে লিটারেল।

এরকম অনেক ক্ষেত্রে আমরা new কিওয়ার্ড ব্যবহার না করেই ভেরিয়েবল initialize করতে পারি।

জাভাতে প্রিমিটিভ টাইপ সকল ডাটাটাইপ লিটারেল সাপোর্ট করে। যেমন -

boolean result = true;

char capitalC = 'C';

byte b = 100;

short s = 10000;

int i = 100000;

নিচে আরো কিছু উদাহরণ দেওয়া হলো –

**ইন্টিজার লিটারেল-**

int decVal = 26; // এখানে 26 হচ্ছে ডেসিমাল নাম্বার

int hexVal = 0x1a; // এখানে 26 সংখ্যাটি হেক্সাডেসিমেল এ দেখানো হয়েছে

int binVal = 0b11010; // এখানে 26 সংখ্যাটি বাইনারি-তে এ দেখানো হয়েছে

**ফ্লােটিং পয়েন্ট লিটারেল-**

double d1 = 123.4;

double d2 = 1.234e2; // একি ভ্যালু বৈজ্ঞানিক উপায়ে লেখা হয়েছে

float f1 = 123.4f;

**ক্যারেক্টার এন্ড স্ট্রিং লিটারেল-**

char এবং String উদ্ধৃতি চিহ্নের ভেতরে লেখা হয়। char ক্ষেত্রে একক উদ্ধৃতি ('') চিহ্ন String এর জন্যে ডবল উদ্ধৃতি ("") চিহ্ন ব্যবহার করতে হয়- যেমন-

char chr = 'A'; // ক্যারেক্টার লিটারেল

String name = "Bazlur"; // স্ট্রিং লিটারের

char এবং String ইউনিকোড ক্যারেক্টার হতে পারে।

আমরা জানি কিভাবে ভেরিয়েবল ইনিশিয়ালাইজ করতে হয়, এবার তাহলে এই ভ্যারিয়বল গুলো দিয়ে কি কাজ করা যায় সেগুলো দেখি।

কোন কাজ করতে হলে একজন কার্যকারী বা অপারেটর লাগে। অপারেটর কিছু অপারেন্ড নিয়ে কাজ করে থাকে তারপর ফলাফল রিটার্ন করে। জাভা প্রোগ্রামিং ল্যাংগুয়েজে বেশ কিছু অপারেটর আছে- সেগুলো দেখা যাক-

**এসাইনমেন্ট অপারেটর (Assignment Operator)**

“=” এটি হচ্ছে এসাইনমেন্ট অপারেটর বাংলায় যাকে বলে সমান সমান চিহ্ন। আমরা একটি Bicycle ক্লাস দেখেছি, এর মাঝে কিছু ভেরিয়েবল দেখেছি-

int cadence = 0;

int speed = 0;

int gear = 1;

এই ভ্যারিয়বল গুলোর ডান পাশে সমান সমান চিহ্নের পর আমরা একটা ভ্যালু বা মান বসিয়েছি। এভাবে আমরা একটি ভ্যারিয়বল এর মাঝে ভ্যালু এসাইন করতে পারি।

**এরিথমেটিক অপারেটর(Arithmetic Operator)**

জাভা প্রোগ্রামিং ল্যাংগুয়েজ-এ যোগ, বিয়োগ, গুন, ভাগ করার জন্যে কিছু অপারেটর আছে। এগুলো আমরা যখন বেসিক গণিত শিখি তখন থেকেই জানি। শুধু একটি অপারেটর নতুন মনে হতে পারে, যা হলো “%”। এটিকে অনেকেই পারসেন্টেজ বা শতকরা চিহ্ণ হিসেবে ভুল করতে পারে, কিন্তু এটি আসলে তা নয় । এটি মূলত একটি সংখ্যাকে আরেকটি সংখ্যা দ্বারা ভাগ করে ভাগশেষ রিটার্ন করে।

|  |  |
| --- | --- |
| অপারেটর | এর কাজ |
| + | আডিটিভ(Additive) অপারেটর, যা দুটি সংখ্যা বা স্ট্রিং যোগ করার জন্যে ব্যবহার করা হয়। |
| - | সাবস্ট্রাকশান (Subtraction) অপরেটর যা একটি সংখ্যা থেকে আরেকটি সংখ্যা বিয়োগ করার জন্যে ব্যবহার করা হয়। |
| \* | মাল্টিপ্লিকেশান (Multiplication)অপারেটর যা দুটি সংখ্যাকে গুন করে। |
| / | ডিভিশান(Division) অপারেটর , যা দিয়ে একটি সংখ্যাকে আরেকটি সংখ্যা দ্বারা ভাগ করা যায়। |
| % | রিমাইন্ডার (Remainder) অপারেটর যা একটি সংখ্যাকে আরেকটি সংখ্যা দ্বারা ভাগ করে ভাগশেষ রিটার্ন করে। |

class ArithmeticDemo {

​

public static void main (String[] args) {

​

int result = 1 + 2; // এখানে result এর মান হচ্ছে 3

System.out.println("1 + 2 = " + result);

int original\_result = result;

​

result = result - 1; //এখানে result থেকে ১ সাবস্ট্রাক্ট করায় এর মান ২

System.out.println(original\_result + " - 1 = " + result);

original\_result = result;

​

result = result \* 2; // এখানে result এর সাথে ২ মাল্টিপ্লাই করার ফলে এর মান 4

System.out.println(original\_result + " \* 2 = " + result);

original\_result = result;

​

result = result / 2; //আবার result ডিভাইড করার ফলে এর মান হয়ে গেল 2

System.out.println(original\_result + " / 2 = " + result);

original\_result = result;

​

result = result + 8; // ৮ যোগ করার ফলে এর result হলো 10

System.out.println(original\_result + " + 8 = " + result);

original\_result = result;

​

result = result % 7;

// result এর সাথে ৭ রিমাইন্ডার অপারেটর ব্যবহার করার ফলে এর মান হয়ে গেল 3, কারণ এটি শুধু মাত্র রিমানইন্ডার বা ভাগশেষ রিটার্ন করে

System.out.println(original\_result + " % 7 = " + result);

}

}

এই প্রোগ্রামটি রান করলে নিচের ফলাফল প্রকাশিত হবে।

1 + 2 = 3

3 - 1 = 2

2 \* 2 = 4

4 / 2 = 2

2 + 8 = 10

10 % 7 = 3

**ইউনারি (Unary) অপারেটর**

উপরের সব অপারেটর এর জন্যে আমাদের দুটি করে অপারেন্ড দরকার হতো, তবে এই অপারেটরের লাগে একটি।

এগুলো বিভিন্ন ধরণের কাজ করে থাকে যেমন – এক করে ইনক্রিমেন্টিং/ডিক্রিমেন্টিং বা একটা এক্সপ্রেশান নেগেট করা বা একটা বুলিয়ান-কে ইনভার্ট করা । এগুলো হল - +, -, ++, –-, ‍! উদাহরণ -

class UnaryDemo {

​

public static void main(String[] args) {

​

int result = +1;

// এটি এক করে ইনক্রিমেন্ট করে, সুতরাং এখানে result এর মান 1

System.out.println(result);

​

result--;

// এটি এক করে ডিক্রিমেন্ট করে, সুতরাং এখানে result এর মান 0

System.out.println(result);

​

result++;

// এটিও এক করে ইনক্রিমেন্ট করে, সুতরাং এখানে result এর মান আবার ১

System.out.println(result);

​

result = -result;

// এখানে result কে নেগেট করে, সুতরাং এর মান এখন -1

System.out.println(result);

​

boolean success = false;

// এখানে বুলিয়ানের মান হচ্ছে false

System.out.println(success);

// কিন্তু এর আগে একটি নেগেট অপারেটর এড করলে এটি হয়ে যায়

System.out.println(!success); // true

}

}

**ইকুয়ালিটি (Equality) এবং রিলেশনাল(Relational) অপারেটরস**

ইকুয়ালিটি (Equality) এবং রিলেশনাল(Relational) অপারেটর গুলো নির্ধারণ করে একটি ভ্যালু অন্যটি থেকে বড় বা ছোট কিনা।

== দুটি ভ্যালু সমান হলে এই এক্সপ্রেশান এর মান true হয়

!= দুটি ভ্যালু সমান না হলে true হয়

> প্রথম ভ্যালু পরের ভ্যালু থেকে বড় হলে true হয়

>= প্রথম ভ্যালু পরের ভ্যালু থেকে বড় বা সমান হলে true হয়

< প্রথম ভ্যালু পরের ভ্যালু থেকে ছোট হলে true হয়

<= প্রথম ভ্যালু পরের ভ্যালু থেকে ছোট বা সমান হলে true হয়

উদাহরণ

‌‌‌‌

class ComparisonDemo {

​

public static void main(String[] args){

int value1 = 1;

int value2 = 2;

if(value1 == value2)

System.out.println("value1 == value2");

if(value1 != value2)

System.out.println("value1 != value2");

if(value1 > value2)

System.out.println("value1 > value2");

if(value1 < value2)

System.out.println("value1 < value2");

if(value1 <= value2)

System.out.println("value1 <= value2");

}

}

কন্ডিশনাল( Conditional) অপারেটর

&& এবং || এই দু্ই অপারেটরকে কন্ডিশনাল অপারেটর বলে।

&& কন্ডিশনাল অ্যান্ড (Conditional-AND)

|| কন্ডিশনাল ওর ( Conditional-OR)

উদাহরণ-

class ConditionalDemo1 {

​

public static void main(String[] args){

int value1 = 1;

int value2 = 2;

if((value1 == 1) && (value2 == 2))

System.out.println("value1 is 1 AND value2 is 2");

if((value1 == 1) || (value2 == 1))

System.out.println("value1 is 1 OR value2 is 1");

}

}