JAVA — 1-BO‘LIM

JAVA ASOSLARI VA SINTAKSIS

1. Nazariy qism

1. Java nima va nega kerak?

Java — bu:

1. kompilyatsiya qilinadigan til
2. platformadan mustaqil (`Write once, run anywhere`)
3. tez va barqaror
4. katta loyihalar va olimpiadalar uchun mos

Olimpiadada Java:

1. Python’dan tezroq
2. C++ ga yaqin ishlash tezligi
3. Xotira xavfsizligi yuqori

2. Java dasturining tuzilishi

Har bir Java dastur:

```
public class Main {
    public static void main(String[] args) {
        // kod shu yerda
    }
}
```

Qoidalar:

1. `class` nomi fayl nomi bilan bir xil
2. `main` — dastur boshlanish nuqtasi
3. `public static void main` → o‘zgarmaydi

3. Ma’lumot turlari (Data Types)

Asosiy (primitive) turlar:

Tur	Hajmi	Misol
`int`	32 bit	`int a = 5;`
`long`	64 bit	`long x = 10L;`

Tur	Hajmi	Misol
`double`	64 bit	`double y = 3.14;`
`char`	16 bit	`char c = 'A';`
`boolean`	1 bit	`true / false`

Olimpiadada:

1. `int` yetarli bo‘lmasa → `long`
2. `double` → ehtiyotkorlik bilan

4. O‘zgaruvchilar va konstantalar

```
int a = 10;
final int MOD = 1000000007;
```

`final` → qiymati o‘zgarmaydi
`MOD` — olimpiada standarti

5. Operatorlar

Arifmetik:

```
+  -  *  /  %
```

Taqqoslash:

```
==  !=  <  >  <=  >=
```

Mantiqiy:

```
&&  ||  !
```

Java’da:

1. `==` → qiymat (primitive)
2. `equals()` → obyektlar (`String`)

6. Shart operatorlari

```
if – else
```

```
if (a > b) {
    System.out.println(a);
} else {
    System.out.println(b);
}
```

`switch`

```
switch (x) {
    case 1: break;
    case 2: break;
    default: break;
}
```

Olimpiadada: `if` → ko‘proq ishlatiladi

7. Sikllar (Loops)

`for`

```
for (int i = 0; i < n; i++) {
    // kod
}
```

`while`

```
while (x > 0) {
    x--;
}
```

Eslatma:

1. `for` → aniq takrorlar
2. `while` → shart asosida

8. Massivlar (Arrays)

E’lon qilish:

```
int[] a = new int[n];
```

Kiritish:

```
a[i] = value;
```

Uzunlik:

```
a.length
```

Java’da:

1. massiv uzunligi o‘zgarmaydi
2. indekslar 0 dan boshlanadi

9. String bilan ishlash

```
String s = "Java";
int len = s.length();
char c = s.charAt(0);
```

Taqqoslash:

```
s.equals(t)
```

`s == t` — xato (ko‘p tushadigan)

10. Kiritish va chiqarish

`Scanner` (sekin)

```
Scanner sc = new Scanner(System.in);
int x = sc.nextInt();
```

`BufferedReader` (tez)

```
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
String line = br.readLine();
```

Olimpiadada:

Scanner — yo‘q,
BufferedReader — ha

11. String → son o‘tkazish

```
int x = Integer.parseInt("123");
long y = Long.parseLong("1000");
```

Juda ko‘p ishlatiladi.

12. Metodlar (Functions)

```
static int sum(int a, int b) {
    return a + b;
}
```

`static` — majburiy (main bilan ishlashi uchun)

13. Paketlar va import

```
import java.util.*;
import java.io.*;
```

Olimpiadada:

```
import java.io.*;
import java.util.*;
```

— standart kombinatsiya

14. Java’da tezlik tushunchasi

Muhim qoidalar:

1. `BufferedReader` + `StringTokenizer`
2. `StringBuilder` → ko‘p chiqarishda
3. `System.out.println` → ehtiyotkorlik bilan

Java sekin emas — noto‘g‘ri yozilsa sekin.

15. Vaqt murakkabligi (asoslar)

Kod	Murakkablik
Bitta sikl	O(n)
Ichma-ich sikl	$O(n^2)$
Binary search	O(log n)

Java’da ham algoritm hal qiluvchi.

1-BO‘LIM NATIJASI

Bu bo‘limni to‘liq o‘zlashtirgan o‘quvchi:

- Java sintaksisini to‘liq tushunadi
- Oddiy va o‘rta masalalarni xatosiz yozadi
- 2-bo‘limga tayyor holatga keladi

JAVA — 1-BO‘LIM

2. METODIK YORDAM (kengaytirilgan)

1. Java’da fikrlashni to‘g‘ri shakllantirish

Metodik qoida:

Java — qat’iy tiplangan til.
Har o‘zgaruvchi turi oldindan aniq.

O‘rgatishda:

1. “keyin o‘ylayman” emas
2. oldindan tip tanlash odatini singdirish

2. Har masalani 3 bosqichda yechish odati

Kiritishni o‘qi (tez usul bilan)
Hisoblash (algoritm)
Chiqishni chiqar

Java’da bu ajratish:

1. kodni toza qiladi
2. xatoni tez topishga yordam beradi

3. `BufferedReader + StringTokenizer` — standart

Metodik qoida:

Olimpiadada `Scanner` taqiqlangan deb hisobla.

Standart shablon:

```
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
StringTokenizer st = new
StringTokenizer(br.readLine());
```

Bu shablon refleks darajasida bo‘lishi kerak.

4. `int` va `long` ni to‘g‘ri tanlash

Metodik qoida:

1. yig‘indi, ko‘paytma → `long`
2. indekslar → `int`

90% WA sababi — `int` overflow.

5. String bilan ishlash metodikasi

1. taqqoslash → `equals()`
2. birlashtirish → `StringBuilder`

```
StringBuilder sb = new StringBuilder();
sb.append(x).append("\n");
```

`+` operatori siklda — yomon odat.

6. Massivlar bilan ishlashda intizom

Metodik urg‘u:

1. indekslar 0 dan
2. `a.length` → metod emas, maydon

`a.length()` — xato!

7. Metod (function) yozish madaniyati

1. kichik vazifalar → alohida metod
2. `static` ni unutmaslik

Juda katta `main` — yomon dizayn.

8. `switch` emas — `if` refleksi

Metodik sabab:

- `switch` kam ishlatiladi
- `if` ko‘proq moslashuvchan

Olimpiadada `if` ustun.

9. Xatoni tez topish strategiyasi

- `System.err.println()` bilan tekshirish
- kichik testlar bilan sinash

To‘liq testda emas — kichik misolda xato qidiriladi.

10. Java tez emas degan stereotipni sindirish

Metodik xulosa:

Java sekin emas — noto‘g‘ri yozilsa sekin.

To‘g‘ri kombinatsiya:

- `BufferedReader`
- `StringTokenizer`
- `StringBuilder`

TIPIK XATOLAR)

1. `Scanner` ishlatish

Katta input → TLE.

2. `String`ni `==` bilan solishtirish

```
if (s == t) // XATO
```

To‘g‘risi:

```
if (s.equals(t))
```

3. `int` overflow

```
int sum = a * b; // XATO bo‘lishi mumkin
```

To‘g‘risi:

```
long sum = 1L * a * b;
```

4. `a.length()` deb yozish

Java’da:

```
a.length   // to‘g‘ri
```

5. `static` ni unutish

`main` ichidan chaqirilgan metod static bo‘lishi shart.

6. `String`ni siklda `+` bilan qo‘shish

Juda sekin ishlaydi.

7. `BufferedReader` dan keyin `Scanner`

Aralash ishlatish → xato.

8. Indeksdan chiqib ketish

```
for (int i = 0; i <= n; i++) // XATO
```

9. `System.out.println`ni haddan ko‘p ishlatish

Katta chiqishda sekinlashadi.

10. Java xatolarini e’tiborsiz qoldirish

Kompilyator ogohlantirishlari — eng yaxshi yordamchi.

METODIK XULOSA

Agar o‘quvchi:

- tez kiritish usullarini bilsa
- tiplarni to‘g‘ri tanlasa
- String va massivlar bilan intizomli ishlasa

Java 1-bo‘lim mustahkam poydevor bo‘ladi.