Përshëndetje të gjithëve!
Sot do të flasim për loops në Java.

Ju do të pyesni, çfarë janë loops?
Ndonjëherë gjatë programimit do të na duhet që një pjesë kodi ta përsërisim disa herë. Nëse këtë pjesë kodi e shkruajmë, le të themi, 10 herë do të na konsumojë më tepër kohë, kodi nuk do të jetë i "pastër" dhe do të jetë më konfuzues. Për këtë na vijnë në ndihmë **loops**. *Loops* na ndihmojnë që një bllok kodi të mund ta përsërisim disa herë pa u dashur që ta ri-shkruajmë. Në Java kemi tre lloje të ndryshme loops; **for loop, while loop** dhe **do-while loop**.

1. For loop

Një for loop i thjeshtë ndërtohet nga katër pjesë:

- **Inicializimi (Initialization):** Është kushti fillestar i cili ekzekutohet një herë kur fillon loop-i. Këtu, mund të deklarojmë një variabël të re, ose mund të përdorim një variabël ekzistuese.
- **Kushti i testimit (Testing Condition):** Ky është kushti i cili kontrollohet për vazhdimin ose mbarimin e loop-it. Duhet të kthejë një vlerë boolean. Nëse vlera është *true* ekzekutimi i loop-it vazhdon, nëse kthen vlerën *false*, atëherë mbaron ekzekutimi.
- **Inkrementimi/Dekrementimi (Increment/Decrement):** Rrit ose zvogëlon vlerën e variablit.
- **Statement:** Ekzekutohet derisa kushti i testimit të bëhet **false**.

Sintaksa e for loop:

```
for(inicializimi; kushti; inkrementimi/dekrementimi) {
    //statement ose kodi që do të ekzekutohet
}
```

Marrim një shembull për printimin e numrave nga 1 deri në 6:

```
for(int i = 1; i <= 6; i++) {
    System.out.println(i);
}
```

Output: 1, 2, 3, 4, 5, 6

Si fillim kemi deklaruar variablin **i** të tipit **int** me vlerë fillestare 1. Më pas kemi kushtin e kontrollit të *loop* (kushtin e testimit) ku kontrollojmë nëse vlera e variablit **i** është <= 6. Në pjesën e tretë kemi inkrementimin me 1 të vlerës së variablit **i**. Brenda bllokut printojmë vlerën e **i** për çdo iterim. Shohim ecurinë e loopit. Në fillim vlera e i = 1 kontrollohet nëse është <= 6 (1 <= 6 kthen true) kështu që vazhdojmë me ekzekutimin e loop-it. Pas këtij kontrolli printohet vlera e parë e **i** (pra, vlera 1). i++ bën të mundur inkrementimin e vlerës së i me 1. Kalojmë në ekzekutimin e dytë të loop-it. Vlera e **i** në këtë rast është ndryshuar nga 1 në 2. Bëhet kontrolli (2 <= 6) i cili kthen vlerën 2 dhe printohet vlera e i (vlera 2). Më pas vlera e i rritet me 1 dhe bëhet 3. Krahasohet përsëri (3 <= 6) e cila kthen përsëri true dhe printohet vlera 3. Kështu vazhdohet derisa vlera e **i** të mos jetë më <= 6 (pra kur të shkojë 7) dhe loop-i do të ndërpritet. Këto hapa tregohen në tabelën më poshtë.

Iterimi	Variabli	Kushti (i <= 6)	Veprimi
1	i = 1	true	Printohet nr 1 Rritet vlera e **i** në 2
2	i = 2	true	Printohet nr 2 Rritet vlera e **i** në 3
3	i = 3	true	Printohet nr 3 Rritet vlera e **i** në 4
4	i = 4	true	Printohet nr 4 Rritet vlera e **i** në 5
5	i = 5	true	Printohet nr 5 Rritet vlera e **i** në 6
6	i = 6	true	Printohet nr 6 Rritet vlera e **i** në 7
7	i = 7	false	Mbyllet loop-i

1.1 Enhanced for loop

Gjithashtu ekziston dhe një **enhanced for loop**. Ky loop përdoret për të iteruar elementët e një **array** ose **collection** në Java. Avantazhi i tij është se eleminon bugs dhe e bën kodin më të lexueshëm. Quhet **for-each loop** pasi kalon çdo element një nga një. Disavantazhi i këtij loop është mos iterimi mbrapsht i elementeve si dhe në këtë loop nuk kemi mundësi të bëjmë *skip* ndonjë element ose të iterojmë vetëm elementët tek/çift, pasi **nuk kemi** një *index*.

Sintaksa e for-each:

```
for(dataType variable : array | collection) {
    //statement ose kodi që do të ekzekutohet
}
```

Marrim një shembull për printimin e numrave në një array.

```
int nums[] = {1, 2, 3, 4, 5, 6};

for(int i : nums) {
    System.out.println(i);
}
```

Si fillim deklarojmë një array me të dhëna të tipit integer. Elementët e tabelës janë nga numri 1 - 6. Më pas printojmë të gjithë numrat e array duke përdorur **for(int i : nums).**

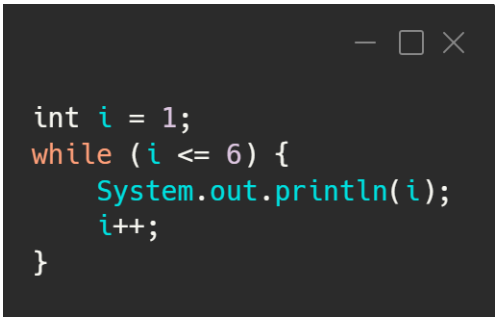
2. While loop

Një **while loop** është një *control flow statement* që lejon që kodi të ekzekutohet në mënyrë të përsëritur bazuar në një kusht të dhënë Boolean. Cikli *while* mund të mendohet si një *if statement* që përsëritet. Në një *while loop*, variabli duhet të inicializohet para se të fillojë loop-i, dhe kjo variabël duhet të përditësohet brenda trupit (body) të while loop.

Sintaksa e while loop:

```
while (testExpression) {
   // body of loop
}
```

Marrim përsëri shembullin e printimit të 6 numrave.



```
int i = 1;
while (i <= 6) {
    System.out.println(i);
    i++;
}
```

While loop fillon duke kontrolluar nëse vlera e variablit i është <= 6. 1 <= 6 kthen ***true*** kështu që vazhdohet me ekzekutimin e bllokut të kodit brenda kllapave. Printohet vlera 1 dhe më pas rritet me 1 vlera e variablit **i**. Kështu me radhë vazhdohet deri në momentin që vlera e variablit **i** bëhet 7 dhe 7 <= 6 kthen false. Këtu loop-i ndalohet.

Iterimi	Variabli	Kushti (i <= 6)	Veprimi
1	i = 1	true	Printohet nr 1 Rritet vlera e **i** në 2
2	i = 2	true	Printohet nr 2 Rritet vlera e **i** në 3
3	i = 3	true	Printohet nr 3 Rritet vlera e **i** në 4
4	i = 4	true	Printohet nr 4 Rritet vlera e **i** në 5
5	i = 5	true	Printohet nr 5 Rritet vlera e **i** në 6
6	i = 6	true	Printohet nr 6 Rritet vlera e **i** në 7
7	i = 7	false	Mbyllet loop-i

3. do-while loop

Do-while është i ngjashëm me while loop, por blloku i kodit ekzekutohet 1 herë para se të kontrollohet kushti. Nëse duam që kodi të ekzekutohet të paktën 1 herë pa patur rëndësi vlera e *testExpression*, atëherë përdorim *do-while*.

Sintaksa e do-while loop:

```
do {
    // body of loop
} while(testExpression);
```

Si fillim ekzekutohet blloku brenda kllapave {} dhe më pas kontrollohet *testExpression* nga **while**.

Marrim përsëri shembullin e printimit të 6 numrave.

```
int i = 1;
do {
    System.out.println(i);
    i++;
} while (i <= 6);
```

Gjatë ekzekutimit të këtij kodi si fillim printohet vlera fillestare e variablit **i** pra, 1. Më pas vlera e variablit rritet me 1 dhe shkon në 2. Bëhet kontrolli nëse (2 <= 6) dhe kthehet vlera boolean *true*. Vazhdon ekzekutimi i trupit të kodit. Printohet vlera 2 dhe rritet vlera e variablit në 3. E kështu me radhë derisa vlera në while të jetë *false* (7 <= 6) dhe loop-i të përfundojë.

Iterimi	Variabli	Kushti (i <= 6)	Veprimi
	i = 1	nuk kontrollohet	Printohet nr 1 Ritet vlera e **i** në 2
1	i = 2	true	Printohet nr 2 Ritet vlera e **i** në 3
2	i = 3	true	Printohet nr 3 Ritet vlera e **i** në 4
3	i = 4	true	Printohet nr 4 Ritet vlera e **i** në 5
4	i = 5	true	Printohet nr 5 Ritet vlera e **i** në 6
5	i = 6	true	Printohet nr 6 Ritet vlera e **i** në 7
6	i = 7	false	Mbyllet loop-i

Përfundim

Në këtë artikull diskutuam 3 llojet e ndryshme të loops në Java që janë for loop, while loop dhe do-while loop. Në tabelën më poshtë jepen të përmbledhura ndryshimet mes tyre.

Krahasimi	For loop	While loop	Do-while loop
Kur përdoret	Nëse numri i iterimeve është i fiksuar rekomandohet përdorimi i for loop	Nëse numri i iterimeve nuk është i fiksuar rekomandohet përdorimi i while loop	Nëse numri i iterimeve nuk është i fiksuar dhe duhet që kodi të ekzekutohet të paktën një herë rekomandohet përdorimi i do-while loop

Sintaksa	for(inicializimi; kushti; inkrementimi/dekrementimi)
	{
	//statement ose kodi që do të ekzekutohet
	}
		while (testExpression) {
		// trupi i loop
		}
			do {
			// trupi i loop
			} while(testExpression)

Shembull	for(int i = 1; i <= 6; i++) {
	System.out.println(i);
	}
		int i = 1;
		while (i <= 6) {
		System.out.println(i);
		i++;
		}
			int i = 1;
			do {
			System.out.println(i);
			i++;
			} while (i <= 6);

Sintaksa për loop infinit	for(;;) {
	//kodi që do të ekzekutohet
	}
		while(true) {
		//kodi që do të ekzekutohet
		}
			do {
			//kodi që do të ekzekutohet
			} while(true);