



Chapter 4

Döngüler (Loops)

Amaçlar

- While döngüsü kullanarak tekrar eden ifadeleri yürütmek için programlar yazmak (§4.2).
- GuessNumber ve SubtractionQuizLoop (§4.2.1) için program geliştirmek.
- Döngü geliştirmek için döngü tasarım stratejisini takip etmek (§4.2.2).
- SubtractionQuizLoop için program geliştirmek (§4.2.3).
- Klavyeden yazmak yerine girdi yönlendirmesini kullanarak dosyadan büyük girdiler elde etmek (§4.2.4).
- While deyimini kullanarak döngüler yazmak (§4.3).
- Do-while deyimini kullanarak döngüler yazmak (§4.3).
- For deyimini kullanarak döngü yazmak (§4.4).
- Üç farklı döngü ifadesinin benzerlik ve farklılıklarını keşfetmek (§4.5).
- İç içe döngüler yazmak (§4.6).
- Sayısal hataları en aza indirme tekniklerini öğrenmek (§4.7).
- Çeşitli örnekler yardımı ile döngüleri öğrenmek (GCD, FutureTuition, MonteCarloSimulation) (§4.8).
- Break ve Continue deyimleri ile program kontrolünü gerçekleştirmek (§4.9).
- (GUI) Onay iletişim kutusuyla bir döngüyü kontrol etmek (§4.10).

Motivasyon

"Java'ya Hoş Geldiniz!" ifadesini yüzlerce kez yazdırmanız gerektiğini varsayalım. Aşağıdaki ifadeyi yüzlerce kez yazmanız sıkıcı olacaktır:

System.out.println("Welcome to Java!");

Bu problemi nasıl çözersiniz?



Açılış Problemi

Problem:

100
defa

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```

...

...

...

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```

While Döngüsü

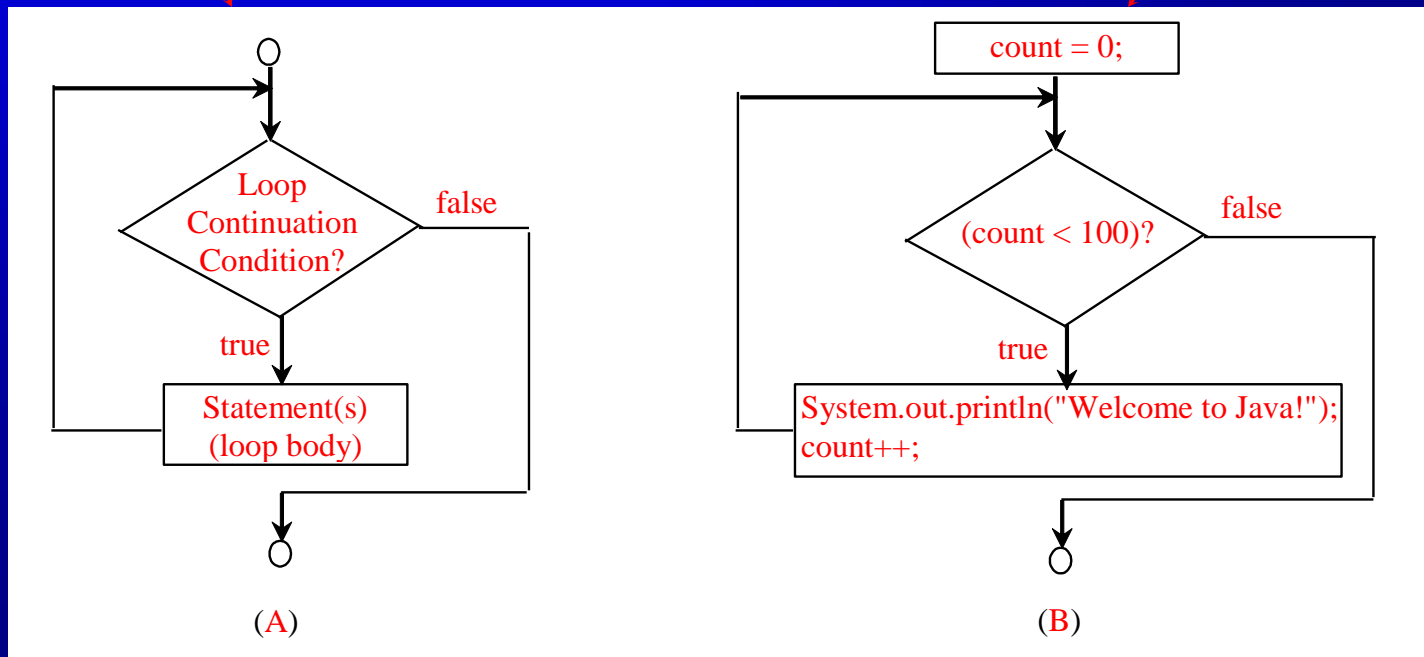
```
int count = 0;  
while (count < 100) {  
    System.out.println("Welcome to Java");  
    count++;  
}
```



While Döngüsü Akış Şeması

```
while (loop-continuation-condition) {  
    // loop-body;  
    Statement(s);  
}
```

```
int count = 0;  
while (count < 100) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```



While Döngüsünün Yürütülmesi

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

Count değişkenini sıfırla



While Döngüsünün Yürütülmesi

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) doğru (true) mu ?
Count değişkenin değeri 0



While Döngüsünün Yürütülmesi

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

«Welcome to Java» ifadesini ekrana yaz



While Döngüsünün Yürütülmesi

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!")  
    count++;  
}
```

count değişkeninin değerini 1 arttır.



While Döngüsünün Yürütülmesi

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

count değişkeni 1 olduğu için
(count < 2) hala doğru (true).



While Döngüsünün Yürütülmesi

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

«Welcome to Java» ifadesini
ekrana yaz



While Döngüsünün Yürütülmesi

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

count değişkeninin değerini 1 arttır.
count değişkeni 2 oldu



While Döngüsünün Yürütülmesi

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) yanlış (false) çünkü
count değeri şu anda 2



While Döngüsünün Yürütülmesi

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

Döngüden çıkılır. Döngüden sonra gelen ilk ifade yürütülür.



Problem: Sayı Tahmini

0-100 arasında rasgele bir tam sayı üreten bir program yazınız. Program, kullanıcıdan rasgele üretilen sayıyı tahmin etmesini isteyecek. Kullanıcı sayıyı girdikten sonra program tahminin sayıdan büyük mü küçük mü olduğunu söyler. Böylelikle, kullanıcı doğru tahmin için yönlendirilmiş olur.

GuessNumberOneTime

Run

GuessNumber

Run

Sayı Tahmini

Program, kullanıcıdan rasgele üretilen sayıyı tahmin etmesini ister. Kullanıcı sayıyı girdikten sonra program tahminin sayıdan büyük mü küçük mü olduğunu söyler. Böylelikle, kullanıcı doğru tahmin için yönlendirilmiş olur.

```
Guess a magic number between 0 and 100
Enter your guess: 50 ↵ Enter
Your guess is too high
Enter your guess: 25 ↵ Enter
Your guess is too low
Enter your guess: 42 ↵ Enter
Your guess is too high
Enter your guess: 39 ↵ Enter
Yes, the number is 39
```



```

1  import java.util.Scanner;
2
3  public class GuessNumber {
4      public static void main(String[] args) {
5          // Generate a random number to be guessed
6          int number = (int)(Math.random() * 101);
7
8          Scanner input = new Scanner(System.in);
9          System.out.println("Guess a magic number between 0 and 100");
10
11         int guess = -1;
12         while (guess != number) {
13             // Prompt the user to guess the number
14             System.out.print("\nEnter your guess: ");
15             guess = input.nextInt();
16
17             if (guess == number)
18                 System.out.println("Yes, the number is " + number);
19             else if (guess > number)
20                 System.out.println("Your guess is too high");
21             else
22                 System.out.println("Your guess is too low");
23         } // End of loop
24     }
25 }

```

	line#	number	guess	output
	6	39		
	11		-1	
iteration 1 {	15		50	
	20			Your guess is too high
iteration 2 {	15		25	
	22			Your guess is too low
iteration 3 {	15		42	
	20			Your guess is too high
iteration 4 {	15		39	
	18			Yes, the number is 39

Döngünün Sentinel Bir Değer ile Bitirilmesi

Genellikle bir döngü yürütme sayısı önceden belirlenmiş değildir. Döngünün sonunu belirtmek için bir giriş değeri kullanabilirsiniz. Böyle bir değer, sentinel değeri olarak bilinmektedir.

```
char continueLoop = 'Y';
while (continueLoop == 'Y') {
    // Execute the loop body once
    ...
    // Prompt the user for confirmation
    System.out.print("Enter Y to continue and N to quit: ");
    continueLoop = input.getLine().charAt(0);
}
```

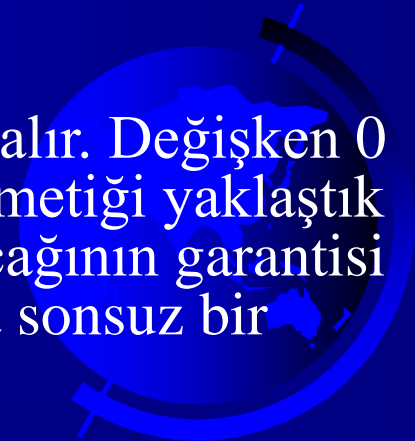
Dikkat !

Döngü kontrolünde eşitlik kontrolü için kayan nokta değerlerini kullanmayın. Kayan nokta değerleri bazı değerlerin yaklaşıkları olduğundan, bunların kullanılması kesin olmayan sayaç değerlerine ve yanlış sonuçlara neden olabilir. $1 + 0.9 + 0.8 + \dots + 0.1$ hesaplamaları için aşağıdaki kodu göz önünde bulundurun:

```
double item = 1; double sum = 0;
while (item != 0) { // No guarantee item will be 0
    sum += item;
    item -= 0.1;
}
```

```
System.out.println(sum);
```

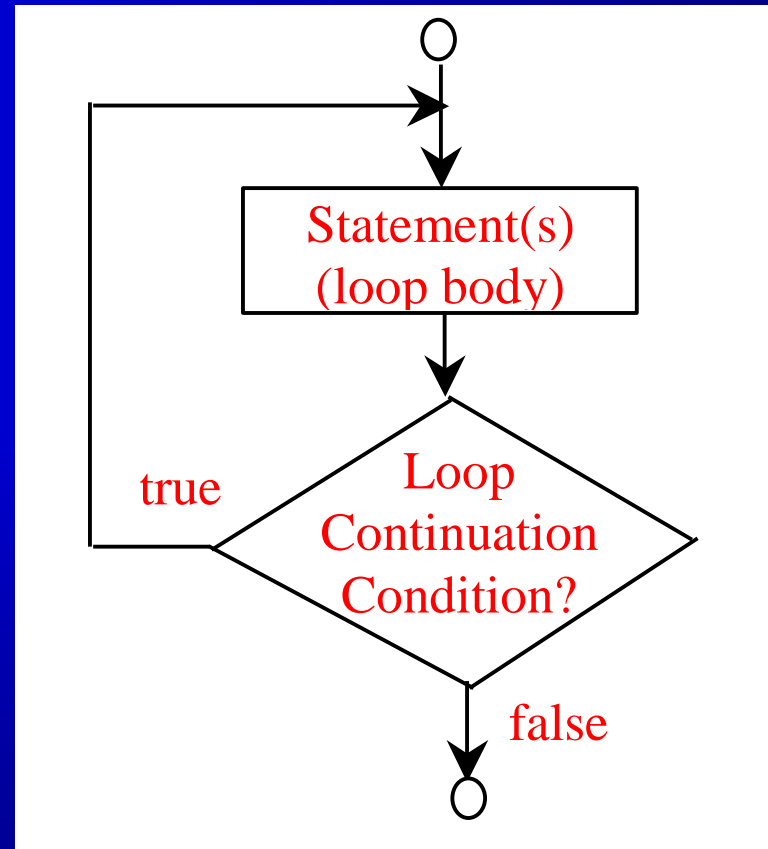
Değişken, 1 ile başlar ve döngü her çalıştırıldığında 0,1 azalır. Değişken 0 olduğunda döngü sona ermelidir. Ancak, kayan nokta aritmetiği yaklaştık oalrak değerlendirildiği için, değişkenin tam olarak 0 olacağının garantisi yoktur. Bu döngü problemsiz gibi görünüyor, ama aslında sonsuz bir döngü.



do-while Döngüsü

Do-while döngüsü while döngüsü ile aynı mantıkla çalışır. Sadece, do-while birkez yürütüldükten sonra döngü durumu kontrol edilir.

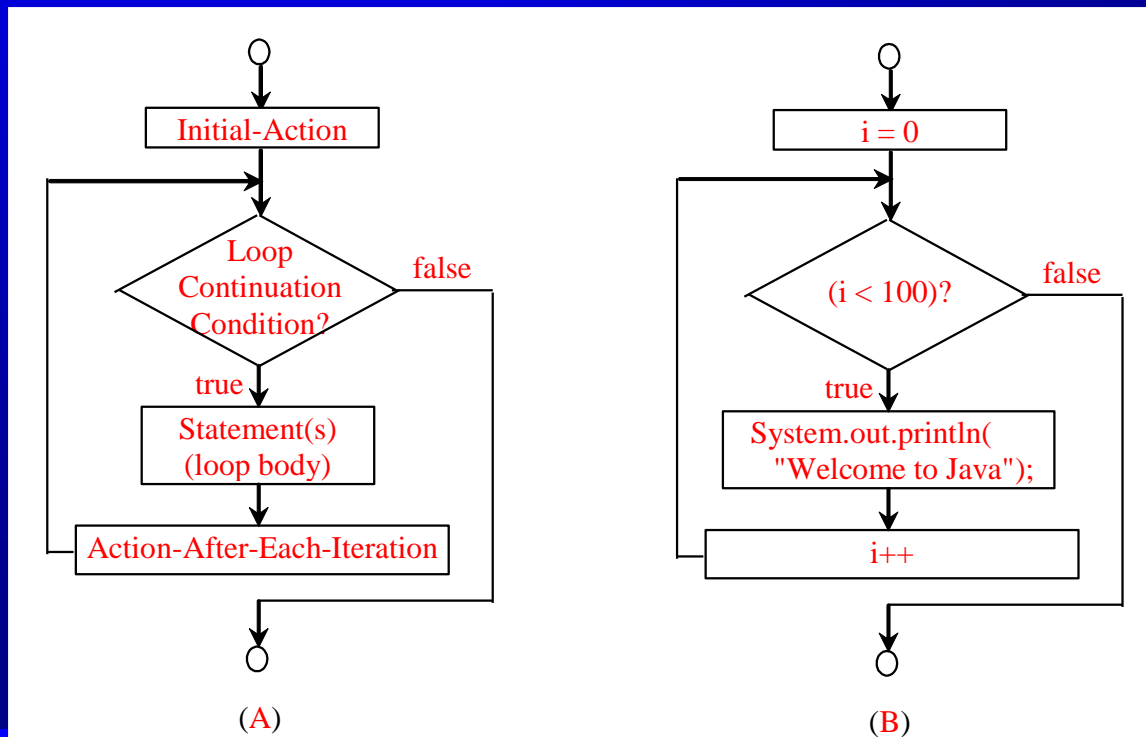
```
do {  
    // Loop body;  
    Statement(s);  
} while (loop-continuation-condition);
```



for Döngüsü

```
for (initial-action; loop-  
    continuation-condition;  
    action-after-each-iteration) {  
    // loop body;  
    Statement(s);  
}
```

```
int i;  
for (i = 0; i < 100; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

i değişkeni deklare edilir.



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Başlatıcıyı yürüt (değişken initialize edilir.) i değişkeninin değeri «0»



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println( "Welcome to Java!");  
}
```

(i < 2) ifadesi true
çünkü i değeri 0



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Welcome to Java yazısı
ekrana yazılır.



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Adjustment (Düzeltilim) ifadesini yürüt, i değeri şimdi 1 oldu.



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

($i < 2$) ifadesi doğru (true). Çünkü i değeri 1.



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Welcome to Java yazısı ekrana yazılır.



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Adjustment (Düzeltilim) ifadesini yürüt, i değeri şimdi 2 oldu.



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

(i < 2) ifadesi false
Çünkü i değeri 2



For Döngüsünün Yürütülmesi

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java");  
}
```

Döngüden çık. Döngüden sonraki ifadeyi yürüt.



Not

Bir for döngüsündeki ilk eylem, sıfır veya daha fazla virgülle ayrılmış ifadelerin bir listesi olabilir. Bir for döngüsündeki her bir yinelemeden sonraki eylem, sıfır veya daha fazla virgülle ayrılmış ifadelerin listesi olabilir. Bu nedenle, döngüler için aşağıdaki iki ifade doğrudur. Ancak, pratikte nadiren kullanılırlar.

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) {  
    // Do something  
}
```



Not

Bir for döngüsündeki döngü devam koşulu çıkarılırsa, ifade dolaylı olarak doğrudur.

Bu nedenle, aşağıda sonsuz bir döngü olan (a) 'da verilen ifade doğrudur. Bununla birlikte, karışıklığı önlemek için (b) 'deki eşdeğer döngüyü kullanmak daha iyidir:

```
for ( ; ; ) {  
    // Do something  
}
```

(a)

Equivalent

```
while (true) {  
    // Do something  
}
```

(b)

Dikkat!

Döngü gövdesinden önce for ifadesinin sonuna noktalı virgül eklemek, aşağıda gösterildiği gibi yaygın bir hatadır:

Logic Error

(Mantıksal Hata)

```
for (int i=0; i<10; i++);  
{  
    System.out.println("i is " + i);  
}
```

Dikkat!

Benzer şekilde, aşağıdaki döngü de yanlıştır:

```
int i=0;  
while (i < 10); ← Logic Error (Mantıksal Hata)  
{  
    System.out.println("i is " + i);  
    i++;  
}
```

Do-while döngüsünde, döngüyü sonlandırmak için aşağıdaki noktalı virgül gereklidir.

```
int i=0;  
do {  
    System.out.println("i is " + i);  
    i++;  
} while (i<10); ← Correct (Doğru)
```



Hangi Döngüyü Kullanmalıyız ?

Döngü ifadelerinin üç şekli, while, do-while ve for birbirleri ile eşdeğerdir, yani, bu üç formdan herhangi birinde bir döngü yazabilirsiniz. Örneğin, aşağıdaki şekilde (a) 'daki bir while döngüsü her zaman (b)' deki döngü için aşağıdakilere dönüştürülebilir:

```
while (loop-continuation-condition) {  
    // Loop body  
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )  
    // Loop body  
}
```

(b)

```
for (initial-action;  
     loop-continuation-condition;  
     action-after-each-iteration) {  
    // Loop body;  
}
```

(a)

Equivalent

```
initial-action;  
while (loop-continuation-condition) {  
    // Loop body;  
    action-after-each-iteration;  
}
```

(b)

Öneriler

Sizin için en sezgisel ve rahat olanı kullanın. Genel olarak, tekrar sayısı biliniyorsa, örneğin 100 kez bir mesaj yazdırmanız gerektiğinde, bir for döngüsü kullanılabilir. Tekrar sayısı bilinmiyorsa, bir while döngüsü kullanılabilir. Örneğin, Girdi 0 olana kadar olan sayıların okunması durumunda while döngüsü kullanılabilir. Devam durumunu test etmeden önce döngü işlemi gerçekleştirilmek istenirse while döngüsünün yerini bir do while döngüsü kullanılabilir



İç İçe Döngüler

Problem: İç içe geçmiş for döngülerini kullanarak çarpım tablosuna ekrana yazdıran programı yazınız.

A screenshot of an IDE window. The top part shows a code editor with the text MultiplicationTable. Below the code editor is a green button with the text "Run". To the right of the code editor and button is a circular graphic with a globe and a curved arrow.

MultiplicationTable

Run

Sayısal Hataların En Aza İndirilmesi

Kayan noktalı sayıları içeren sayısal hatalar kaçınılmazdır. Bu bölümde, bir örnek aracılığı ile bu tür hataların nasıl en aza indirileceğini tartışacağız.

TestSum

Run

Problem: En Büyük Ortak Bölenin Bulunması

Problem: Kullanıcıdan iki pozitif tamsayı girmesini isteyen ve bu sayıların en büyük ortak bölenini bulan bir program yazın.

Solution: İki tam sayı 4 ve 2'yi girdiğinizi varsayalım, bu sayıların en büyük ortak böleni 2'dir. İki tam sayı 16 ve 24'ü girdiğinizi varsayalım bu sayıların en büyük ortak böleni 8' dir. Peki, en büyük ortak böleni nasıl bulursunuz? İki giriş tamsayısının $n1$ ve $n2$ olmasını sağlayın. 1 sayısının ortak bir bölen olduğunu biliyorsunuz, ama en büyük ortak bölen olmayabilir. K 'nin ($k = 2, 3, 4$ vb. için) $n1$ veya $n2$ 'den büyük olana kadar $n1$ ve $n2$ için ortak bir bölen olup olmadığını kontrol edebilirsiniz.

GreatestCommonDivisor

Run

Problem: Gelecekteki Öğrenim Ücretini Tahmin Etmek

Problem: Bir üniversite öğreniminin bu yıl 10,000 dolar olduğunu ve her yıl öğrenimin% 7 arttığını varsayalım. Eğitim ücreti kaç yıl sonra iki katına çıkacaktır?

FutureTuition

Run

Problem: Gelecekteki Öğrenim Ücretini Tahmin Etmek

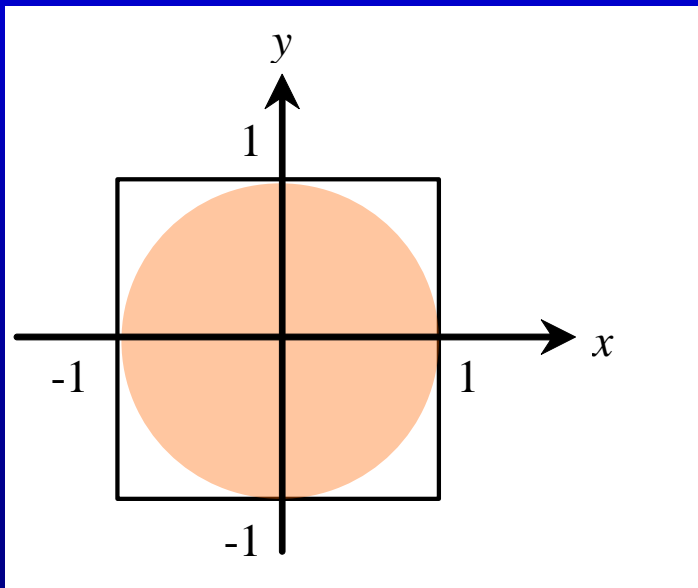
```
double tuition = 10000; int year = 1 // Year 1
tuition = tuition * 1.07; year++;      // Year 2
tuition = tuition * 1.07; year++;      // Year 3
tuition = tuition * 1.07; year++;      // Year 4
...
```

FutureTuition

Run

Problem: *Monte Carlo Simülasyonu*

Monte Carlo simülasyonu, problemleri çözmek için rasgele sayılar ve olasılık kullanan bir tekniği ifade eder. Bu yöntem hesaplamalı matematik, fizik, kimya ve finans alanlarında geniş bir uygulama alanına sahiptir. Bu bölüm, π değerinin tahmini için Monte Carlo simülasyonunun kullanılmasını örneklemektedir.



$$\frac{\text{circleArea}}{\text{squareArea}} = \pi / 4.$$

$$\pi \text{ can be approximated as } \frac{4 * \text{numberOfHits}}{1000000}.$$

MonteCarloSimulation

Run

Break ve Continue Komutlarının Kullanılması

break and continue anahtar kelimelerini
kullanmak için örnekler:

□ TestBreak.java

TestBreak

Run

□ TestContinue.java


TestContinue

Run



TestBreak.java


```
1  public class TestBreak {
2      public static void main(String[] args) {
3          int sum = 0;
4          int number = 0;
5
6          while (number < 20) {
7              number++;
8              sum += number;
9              if (sum >= 100)
10                 break;
11         }
12
13         System.out.println("The number is " + number);
14         System.out.println("The sum is " + sum);
15     }
16 }
```



```
The number is 14
The sum is 105
```

TestContinue.java

```
1  public class TestContinue {
2      public static void main(String[] args) {
3          int sum = 0;
4          int number = 0;
5
6          while (number < 20) {
7              number++;
8              if (number == 10 || number == 11)
9                  continue;
10             sum += number;
11         }
12
13         System.out.println("The sum is " + sum);
14     }
15 }
```



The sum is 189

Sayı Tahmini Probleminin Tekrardan Ele Alınması

Break ifadesini kullanarak sayı tahmin programını yeniden yazabilirsiniz.

GuessNumberUsingBreak

Run

Problem: Asal Sayıları Görüntüleme

Problem: Her biri 10 sayı içeren beş satırda ilk 50 asal sayıyı gösteren bir program yazınız. Tek pozitif böleni 1 veya kendisi olan, 1'den büyük bir tamsayı asaldır. Örneğin, 2, 3, 5 ve 7 asal sayılardır, ancak 4, 6, 8 ve 9 asal sayı değildir.

Çözüm: Problem aşağıdaki işlem adımlarına bölünebilir.

- Sayı = 2, 3, 4, 5, 6, ... için, sayının asal olup olmadığını deneyin.
- Verilen sayının asal sayı olup olmadığına karar verin.
- Asal sayıları yazın.
- Her asal sayıyı yazdırın ve her satıra 10 sayı yazdırın.

PrimeNumber

Run