סיכום Java

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Description** |
| byte | 1 byte | numbers from -128 to 127 |
| short | 2 bytes | numbers from -32,768 to 32,767 |
| int | 4 bytes | numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 bytes | Sufficient for storing 6 to 7 decimal digits |
| double | 8 bytes | Sufficient for storing 15 decimal digits |
| boolean | 1 bit | true or false values |
| char | 2 bytes | Stores a single character/letter or ASCII values |

מחלקת Math.

|  |  |  |  |
| --- | --- | --- | --- |
| **הפעולה** | **טיפוס אפשרי** | **טיפוס מוחזר** | **תיאור** |
| Math.abs | שלם/ממשי | שלם/ממשי | ערך מוחלט |
| Math.sqrt | ממשי | ממשי | שורש ריבועי |
| Math. Pow | ממשי,ממשי | ממשי | חזקה |
| Math.min | שלם/ממשי | שלם/ממשי | מוצא מינימום |
| Math.max | שלם/ממשי | שלם/ממשי | מוצא מקסימום |
| Math.round | ממשי | שלם | מעגל למעלה |

**כיצד להשתמש ב Scanner וב Random:**

import java.util.Scanner;

import java.util.Random;

Scanner in = new Scanner(System.in);

Random rnd = new Random();

**קליטת קלט מהמשתמש:**

קליטת מחרוזת – in.next()

קליטת תו – in.next().charAt(0)

קליטת מספר שלם – in.nextInt()

קליטת מספר ממשי – in.nextDouble() או in.nextFloat()

קליטת מספר שלם ארוך – in.nextLong()

**הדפסת פלט למשתמש:**

הדפסה באותה שורה – System.out.print()

הדפסה בשורה חדשה – System.out.println

**חיבור ספרות במספר שלם:**

int sum = 0;

while (num > 0) {

Sum = sum + (num % 10);

num = num / 10;

}

**חישוב כמות התווים במספר שלם:**

int i = 0;

while (nun > 0) {

num = num / 10;

i++;

}

System.out.println(i);

**באיזו לולאה נשתמש:**

for – כאשר ידוע לנו אורך הלולאה.

while – כאשר אורך הלולאה אינו ידוע.

do while – כאשר נרצה שהקטע קוד יתבצע לפחות פעם אחת.

**סוגי לולאות:**

for – בצע num פעמים.

for (;;) – לולאה אין סופית.

while – כל עוד (תנאי בוליאני מתקיים), בצע.

do while – דומה לwhile עם הבדל שהתנאי הבוליאני נבדק רק אחרי ביצוע הקוד, מה שיגרום ללפחות ביצוע אחד של הקוד.

|  |
| --- |
| for (init; condition; step) =>  for (int i = 0; i = 9; i++) {  // statements  } |
| while (condition) {  // statements  } |
| do {  // statements  } while (condition); |

**משולשים:**

|  |  |
| --- | --- |
| \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\* | for (int i = 1; i <= 6; i++) {  for (int j = 0; j < 6 - i; j++)  System.out.print(" ");  for (int j = 0; j < i \* 2 - 1; j++)  System.out.print("\*");  System.out.println();  } |
| \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\* | for (int i = 1; i <= 5; i++) {  for (int j = 1; j <= i; j++)  System.out.print("\*");  System.out.println();  } |
| \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\* | for (int i = 1; i <= 5; i++) {  for (int j = 1; j <= 5 - i; j++)  System.out.print(" ");  for (int k = 1; k <= i; k++)  System.out.print("\*");  System.out.println();  } |
| \*\*\*\*\*  \*\*\*\*  \*\*\*  \*\*  \* | for (int i = 1; i <= 5; i++) {  for (int j = 1; j <= i; j++)  System.out.print(" ");  for (int k = i; k <= 5; k++)  System.out.print("\*");  System.out.println();  } |

**פונקציה סטטית לחישוב סכום ספרות:**

public static int sumDigits (int num) {

int sum = 0;

num = Math.abs(num);

while(num > 0) {

sum += num % 10;

num = num / 10;

}

return sum;

}

**פונקציה סטטית להיפוך מספר:**

public static long reverse (int num) {

long reversed = 0;

int digit;

num = Math.abs(num);

while (num > 0) {

digit = num % 10;

reversed = reversed \* 10 + digit;

num = num / 10;

}

return reversed;

}

**פונקציה סטטית לשילוב מעורבב בין שני מספרים:**

public static long merge (int num1, int num2) {

long merged = 0;

long reversedNum1 = reverse(num1\*10+1);

long reversedNum2 = reverse(num2\*10+1);

while (reversedNum1 > 9 || reversedNum2 > 9) {

if (reversedNum1 > 9) {

merged = merged \* 10 + reversedNum1 % 10;

reversedNum1 = reversedNum1 / 10;

}

if (reversedNum2 > 9) {

merged = merged \* 10 + reversedNum2 % 10;

reversedNum2 = reversedNum2 / 10;

}

}

return merged;

}

**חישוב עצרת לולאת for:**

for (int i = 1; i <= num; i++)

factorial \*= i;

**חישוב עצרת לולאת while:**

int i = 1;

while (i < num) {

factorial \*= i;

i++;

}

**חישוב סכום שברים:**

for (int i = 1; i <= num; i++)

sum = sum + (1 / (double) i);

**כימוס:**

public – מאפשר גישה למידע ממחלקות אחרות.

private – מבטל גישה למידע ממחלקות אחרות.

**פונקציה סטטית:** לא נדרש אובייקט ע"מ לקרוא לפונקציה. (מילת מפתח: static).

**מתודה:** נדרש אובייקט ע"מ לקרוא לפונקציה.

**נוסחאות מתמטיות**

**פרבולה:**

חישוב Y:

public double calculateY(double x) {

return (a \* x \* x) + (b \* x) + c;

}

חישוב דיסקרימיננטה, כאשר קטנה מ0 אין תשובה,

כאשר שווה ל0 יש תשובה אחת, כאשר גדולה מ0 יש שתי תשובות:

public double discriminant() {

double x1, x2, delta;

delta = (b \* b) - (4 \* a \* c);

}

חישוב נקודת קיצון, משתמש במחלקת Point:

public Point getMinMax() {

double x = -b / (2 \* a);

double y = (a \* x \* x) + (b \* x) + c;

return new Point(x, y);

}

**מלבן:**

חישוב היקף: (צלע + צלע) \* 2.

חישוב שטח: צלע \* צלע.

חישוב אורך צלע מנקודות, כאשר ה – y זהה: יש להחסיר את ערכי ה – x של שתי הנקודות.

חישוב אורך צלע מנקודות, כאשר ה – x זהה: יש להחסיר את ערכי ה – y של שתי הנקודות.

public double calcLenAB() {

return Math.abs(a.getX()-b.getX());

}

public double calcLenBC() {

return Math.abs(b.getY()-c.getY());

}

**עיגול:**

חישוב שטח: פאי \* רדיוס בריבוע.

חישוב היקף: 2 \* פאי \* רדיוס.

double Sring = (ring\*ring\*Math.PI); - S

**משולש:**

חישוב היקף: חיבור של כל הצלעות המשולש.

חישוב שטח על פי נוסחת הרון:

S = (ab + bc + ac ) / 2;

Area = Math.sqrt(s\*(s-ab)\*(s-bc)\*(s-ac);

**מהו טיפוס?**

**טיפוס הוא תבנית המאפיינת עצמים –**

מבנה מחלקה המייצגת טיפוס –

Public class <שם הטיפוס>

//תכונות

//פעולות בונות

//פעולות פנימיות

**כותרת המחלקה** –

שם המחלקה מתחיל במילים public class ולאחריו שם המחלקה, נהוג ששם המחלקה מתחיל באות גדולה.

שם הקובץ בו נשמרת המחלקה צריך להיות זהה לשם המחלקה.

**הגדרת תכונות –**

הגדרה של תכונה כוללת את הרשאת הגישה לתכונה, את הטיפוס ואת שם התכונה.

בשלב זה כל התכונות תוגדרנה עם בקרת גישה private.

הבקרה מאפשרת גישה ישירה לערכי התכונות **רק** בתוך המחלקה בה הוגדרו.

**הגדרת פעולות בונות –**

פעולה בונה היא פעולה המגודרת בכל טיפוס ובלעדיה לא ניתן לבנות עצמים.

פעולה זו תופעל רק פעם אחת עבור כל עצם.

Public שם\_הפעולה\_כשם\_הטיפוס(שמות הפרמטרים וטיפוסיהם)

//הוראות לביצוע

**מבנה פעולה בונה –**

לפעולה בונה יש שם זהה לשם המחלקה המייצגת את הטיפוס.

**ביצוע פעולה בונה כוללת ארבעה שלבים :**

שלב ראשון – הגדרת שטחי זיכרון מתאימים לעצם החדש על פי הגדרת התכונות בטיפוס שלו.

שלב שני – אתחול התכוונת על פי ערכי ברירת המחדל של הטיפוסים שלהן.

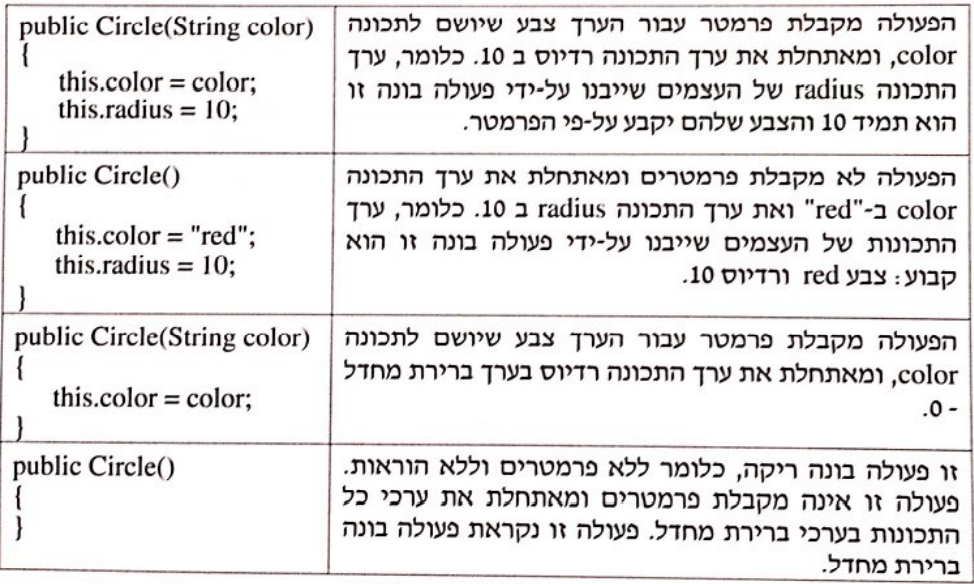
שלב שלישי – ביצוע גוף הפעולה הבונה.

שלב רביעי – החזרת העצם למקום ממנו זומנה הפעולה הבונה.

הפעולה הבונה קובעת ערכים לתכונות העצם.

כלומר הפעולה הבונה יכולה לקבל פרמטר המכיל ערך עבור כל תכונה שאת ערכה היא רוצה לקבוע. ערכי התכונות יכולים להתקבל גם ע"י השמה של קבוע לתכונה. אם ערכי התכונות לא נקבעו על פי פרמטרים או על פי קבועים נשאר בהם הערך של ברירת מחדל לפי הטיפוס של התכונה שהושם בשלב 2.

כדי להבחין בין משתנה הפרמטר למשתנה התכונה , משתמשים במילה השמורה this המציינת שהכוונה לתכונה של העצם הנוכחי אליו מתייחסת הפעולה.

**אם נרצה בנאי שכולו יקבע ע"פ הפרמטרים:**

Public circle(String color , int radius);

{

This.color = color;

This.radius = radius;

}

זה אינו בנאי דיפולטיבי, הוא מקבל את ערך 0 ברדיוס ואת הערך null בצבע כיוון והוא string.