The document is in Hebrew and contains code examples and explanations related to programming in Java. The text includes Java code snippets and explanations of type conversion in Java. The code examples are written in Java and demonstrate type conversion between different data types such as byte, short, int, float, and double. The explanations are written in Hebrew and provide a detailed description of how type conversion works in Java, including narrowing, widening, and the implications of these conversions on variable types.

The code examples are placed within the document as follows:

```java
// Demonstrating narrowing and widening
private static void wideningExample() {
    // Narrowing
    int i = 123;
    long l = i;
    System.out.println(l); // 123

    // Widening
    float f = 123.456f;
    double d = f;
    System.out.println(d); // 123.456
}
```

The text also includes a discussion on the implications of type conversion on variable types and the importance of considering these conversions in code to avoid potential runtime errors or performance issues. The document is likely intended for readers familiar with Java programming and looking to deepen their understanding of type conversion in the language.
private static void wideningExample() {
    float fmin = Float.NEGATIVE_INFINITY;
    float fmax = Float.POSITIVE_INFINITY;
    printIn("long: " + (long) fmin + ";" + (long) fmax);
    printIn("int: " + (int) fmin + ";" + (int) fmax);
    printIn("char: " + (char) fmin + ";" + (char) fmax);
    printIn("byte: " + (byte) fmin + ";" + (byte) fmax);
}

private static void wideningExample() {
    float fmin = Float.NEGATIVE_INFINITY;
    float fmax = Float.POSITIVE_INFINITY;
    printIn("long: " + (long) fmin + ";" + (long) fmax);
    printIn("int: " + (int) fmin + ";" + (int) fmax);
    printIn("char: " + (char) fmin + ";" + (char) fmax);
    printIn("byte: " + (byte) fmin + ";" + (byte) fmax);
}

public static void main(String[] args) {
    int l = 1000000000;
    int i = (int) 1.999999999;
    float f = 5.2f;
    if (f == 3)  // f = 3
    f = (float) (5/2);  // f = 2
    f = (float) 5/2;  // f = 2.5
    f = 5 / (float) 2;  // f = 2.5
    short s = 2;
    if (short a = 2) {  // "Why is this ok?"
        short c = a*4;
    }  // compilation error: cannot convert int to short
}

Interning

Mcchein Shemrhorot

모든 문자열

`String[] array = new String[1000];
for (int i = 0; i < array.length; i++) {
    array[i] = "Hello world ";
}
array

... An immutable string. Thus, can be shared.

Example for - Interning

```
String hello = "Hello", lo = "lo";
System.out.println(hello == "Hello");
System.out.println(Other.hello == hello);
System.out.println(hello == ("Hel"+"lo");
System.out.println(hello == ("Hel"+lo);
System.out.println(hello == ("Hel"+lo).intern());
```

Literal strings within the same class represent references to the same String.

Literal strings within different classes represent references to the same String object.

Strings computed by constant expressions are computed at compile time and then treated as if they were literals.

Strings computed by concatenation at run time are newly created and therefore distinct.
String hello = "Hello", lo = "lo";
System.out.println(hello == "Hello");
System.out.println(Other.hello == hello);
System.out.println(hello == ("Hel"+'lo'));
System.out.println(hello == ("Hel"+lo));
System.out.println(hello == ("Hel"+lo).intern());

Example for Interning
Explicitly interning a String returns a reference to the interned String object. If such a String was previously interned the returned value will refer to that object.

static method

A static method (a class method) is a method of a class that is called by a class name. A static method can access static class variables.

static example

class Point {
    int x, y, useCount;
    Point(int x, int y) { this.x = x; this.y = y; }
    static Point origin = new Point(0, 0);
}

class Test {
    public static void main(String[] args) {
        Point p = new Point(1, 1);
        Point q = new Point(2, 2);
        p.x = 3;
        p.y = 3;
        p.useCount++;
        p.origin.useCount++;
        System.out.println("(\" + q.x + ",\" + q.y + ");
        System.out.println(q.useCount);
        System.out.println(q.origin == Point.origin);
        System.out.println(q.origin.useCount);
    }
}

final fields

A final field can only be assigned a value during the declaration. After that, the field cannot be changed. It is used to ensure that certain fields are always initialized.

static field

A static field (a class variable) is a field of a class. It is shared by all instances of the class. An instance variable (a field of an instance) is private to the instance and cannot be accessed by other instances.