

Java Assignment4
on Datatype conversion

YANATI RAJANIVAS

CONTACT

- ✉ rajanivas9573@gmail.com
- ☎ +919573627556
- 🌐 <http://www.linkedin.com/in/yanati-rajanivas-96004232>
- 📍 1-164 Kalavakonda, Chillakur Mandal(%24412), Tirupati Diostriect.

SKILLS

- Python, Java Basics
- Html and Css
- Javascript
- Sql
- Database Fundamentals

EDUCATION

- **Bachelor Of Technology (6.8 CGPA)**
NBKR Institute of Science and Technology
2019 - 2023
Vidyanagar,Nellore
Computer Science Engineering
- **INTERMEDIATE (9.1 CGPA)**
NBKR Science And Arts College
2017 - 2019
Vidyanagar,Nellore
MPC
- **S.S.C (8.2 CGPA)**
Z.P.P High School
2016 - 2017
Thikkavaram, Nellore

LANGUAGES

- English 
- Telugu 

CAREER OBJECTIVE

Highly motivated and enthusiastic to move in software industry as a graduate seeking an opportunity to apply my knowledge, skills, and passion in a dynamic work environment. Eager to contribute to organization by utilizing my technical knowledge and gaining practical experience in work. Committed to continuous learning and growth, with a strong dedication to achieving professional excellence.

PROJECTS AND INTERNSHIPS

Projects

- **Design Of secured Authenticated Key Management Protocol For Cloud computing Environments**
Front End : HTML , CSS, Javascript Back End: JSP
Data base: MYSQL 5.0 Server: Apache Tomcat

Internships

- **Web Development using Html and Css**
company : Think -Champ pvtLtd Duration: 1 Month
Project : Flower Shop Through online
- **Completed internship in machine learning with python**
company : Verzeo Edutech Duration: 1 Month
Project: Heart Disease prediction using Machine Learning

ACHIEVEMENTS

- ✓ Second place winner in district level drawing competition
- ✓ Won second prize in ball batmenton game under district level

STRENGTHS

- Research and Analysis
- Quick learner
- Positivity and Adaptability
- Team Player

CERTIFICATIONS

- ❖ Python
- ❖ Html And Css
- ❖ Machine Learning
- ❖ Azure AI Fundamentals
- ❖ Database Administrator Fundamentals

1) Char to Char Datatype Conversion:

In Java, you don't need explicit type casting when converting a `char` to another `char`, as they are both of the same data type. A `char` represents a single 16-bit Unicode character, and it can be assigned to another `char` variable directly.

Result:

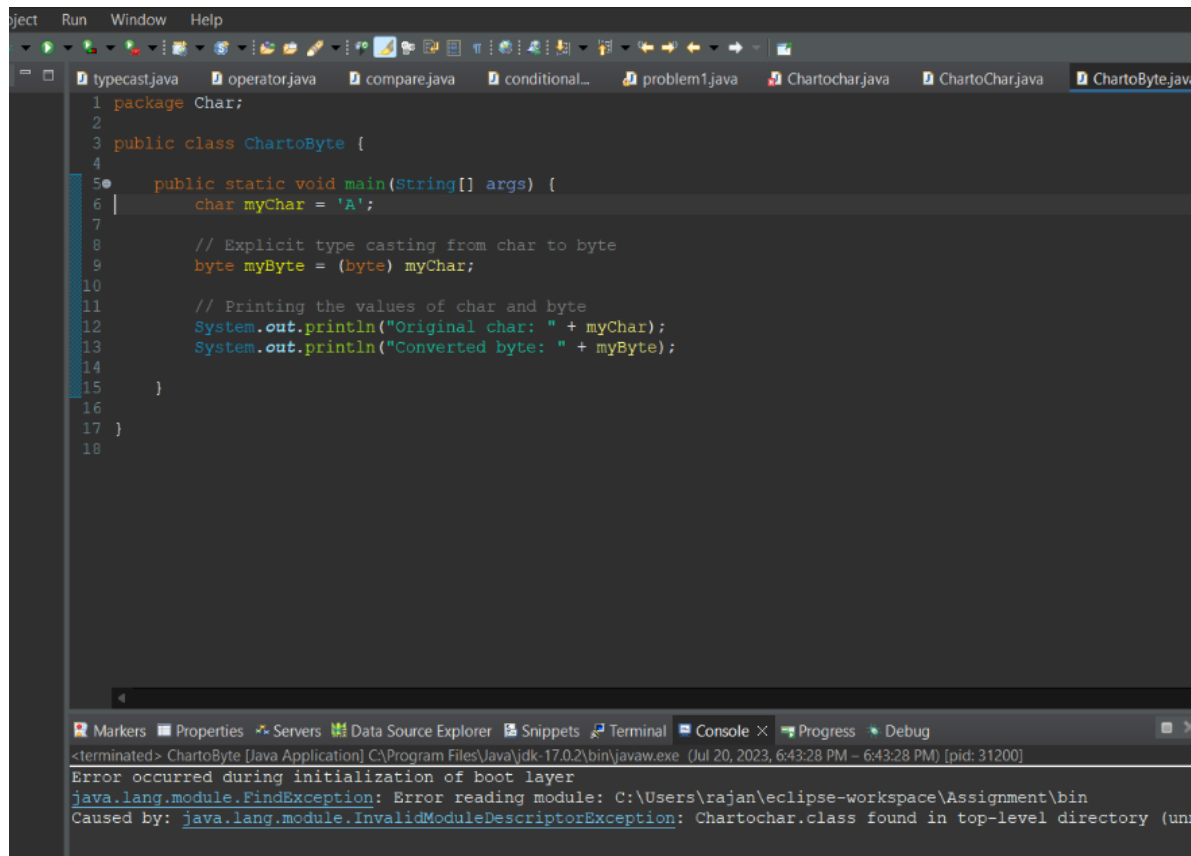
No conversion required for char to char.

2) Char to Byte Datatype Conversion:

In Java, you can perform explicit type casting to convert a `char` to a `byte`. The `char` data type represents a 16-bit Unicode character, whereas the `byte` data type is an 8-bit signed integer.

Since a `char` requires 16 bits to store a character, and a `byte` can only store 8 bits, you need to be cautious when performing this type casting. If the `char` value being casted is outside the valid range of a `byte`, it may result in data loss or unexpected behavior.

Example:



```
object Run Window Help
typecast.java operator.java compare.java conditional... problem1.java Chartochar.java CharToChar.java CharToByte.java
1 package Char;
2
3 public class CharToByte {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Explicit type casting from char to byte
9         byte myByte = (byte) myChar;
10
11        // Printing the values of char and byte
12        System.out.println("Original char: " + myChar);
13        System.out.println("Converted byte: " + myByte);
14    }
15 }
16
17 }
18

Markers Properties Servers Data Source Explorer Snippets Terminal Console X Progress Debug
<terminated> CharToByte [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 6:43:28 PM - 6:43:28 PM) [pid: 31200]
Error occurred during initialization of boot layer
java.lang.module.FindException: Error reading module: C:\Users\rajan\eclipse-workspace\Assignment\bin
Caused by: java.lang.module.InvalidModuleDescriptorException: Chartochar.class found in top-level directory (un
```

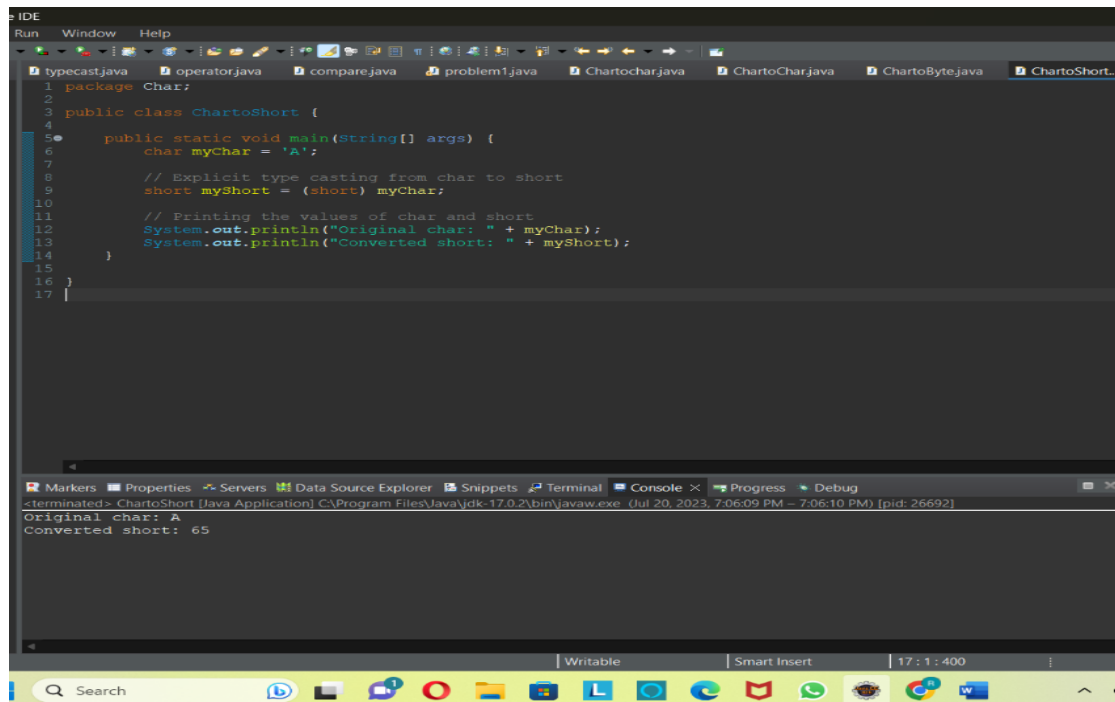
Output:

Error Occurred and it may result in data loss because it is explicit type conversion.

3)Char to Short Data Conversion:

In Java, It needs to perform explicit type casting to convert a `char` to a `short`. The `char` data type represents a 16-bit Unicode character, whereas the `short` data type is a 16-bit signed integer.

Example:



```
IDE
Run Window Help
typecast.java operator.java compare.java problem1.java CharToChar.java CharToChar.java CharToByte.java CharToShort...
1 package Char;
2
3 public class CharToShort {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Explicit type casting from char to short
9         short myShort = (short) myChar;
10
11         // Printing the values of char and short
12         System.out.println("Original char: " + myChar);
13         System.out.println("Converted short: " + myShort);
14     }
15 }
16
17
Markers Properties Servers Data Source Explorer Snippets Terminal Console X Progress Debug
<terminated> CharToShort [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:06:09 PM - 7:06:10 PM) [pid: 26692]
Original char: A
Converted short: 65
Writable Smart Insert 17:1:400
```

Result:

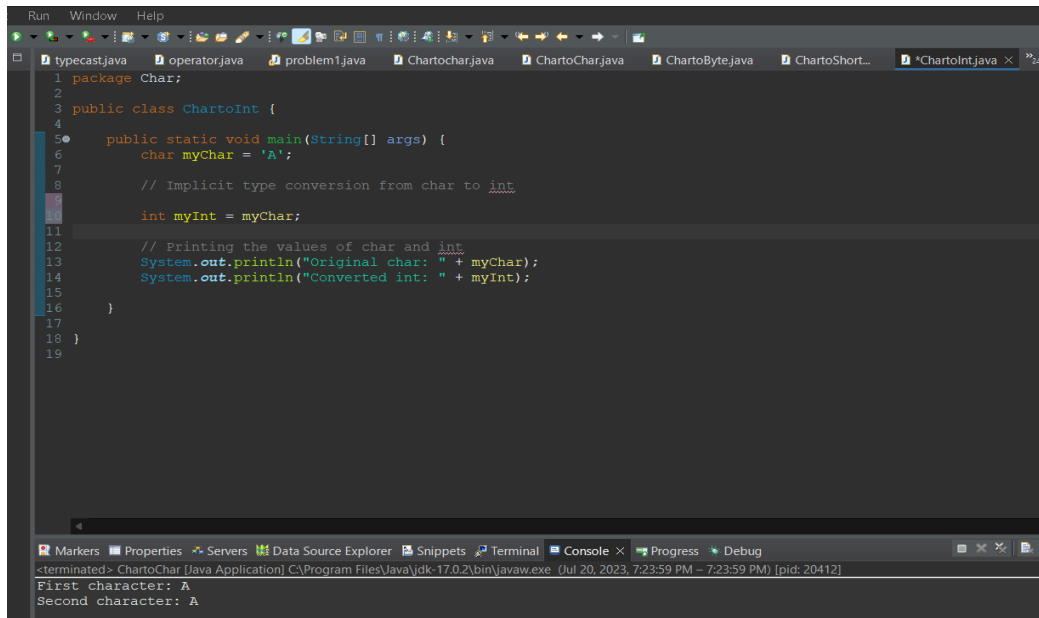
It is done explicit conversion both `char` and `short` are 16-bit data types in Java, so the type casting can be done directly without any loss of data or precision.

4) Char to Int Data conversion:

In Java, you can perform an implicit type conversion from `char` to `int` because `char` is a smaller data type than `int`. The `char` data type represents a 16-bit Unicode character, whereas the `int` data type is a 32-bit signed integer.

When converting a `char` to an `int`, the Unicode code point of the character is used as the integer value.

Example:



```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19
```

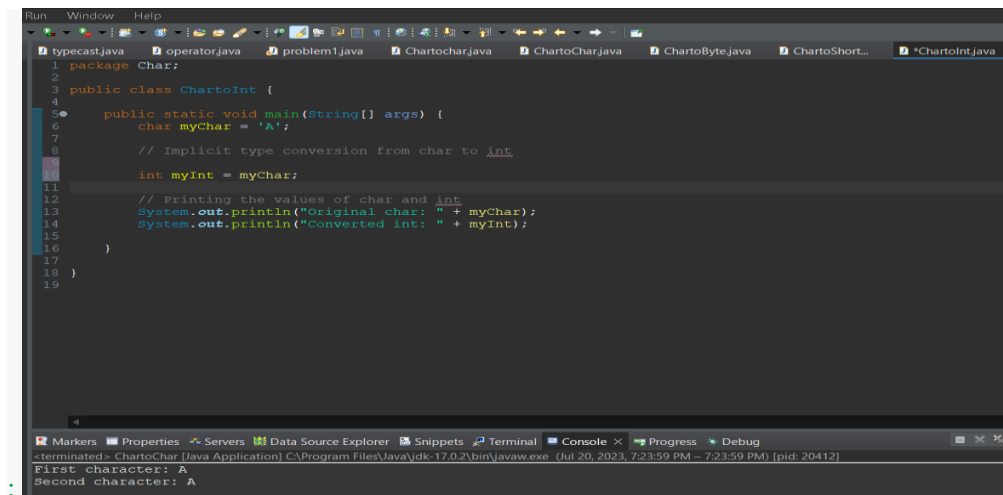
First character: A
Second character: A

5) Char to Long Type Conversion:

In Java, you can perform an implicit type conversion from **char** to **long**. The **char** data type represents a 16-bit Unicode character, whereas the **long** data type is a 64-bit signed integer.

where data is converted from a smaller data type (**char**) to a larger data type (**long**) without any data loss.

Example



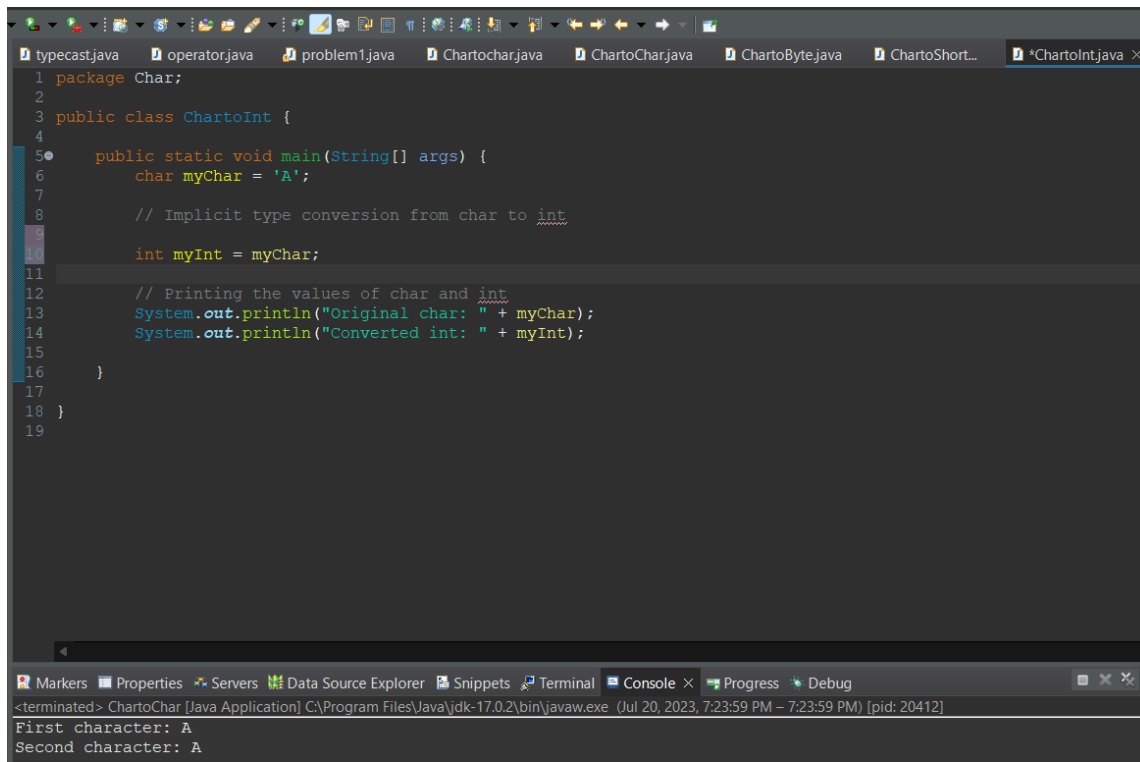
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19
```

First character: A
Second character: A

6) Char to Float conversion:

In Java, you can perform an implicit type conversion from **char** to **float**. The **char** data type represents a 16-bit Unicode character, whereas the **float** data type is a 32-bit single-precision floating-point number.

Example:

A screenshot of an IDE window showing a Java file named 'CharToInt.java'. The code defines a package 'Char' and a public class 'CharToInt'. Inside the class, there is a 'main' method that takes a 'String[] args' parameter. It declares a 'char myChar' and assigns it the value 'A'. A comment indicates an implicit type conversion from 'char' to 'int'. Then, it declares an 'int myInt' and assigns it the value of 'myChar'. Finally, it prints the original 'char' and the converted 'int' to the console. The IDE's console window at the bottom shows the output: 'First character: A' and 'Second character: A'.

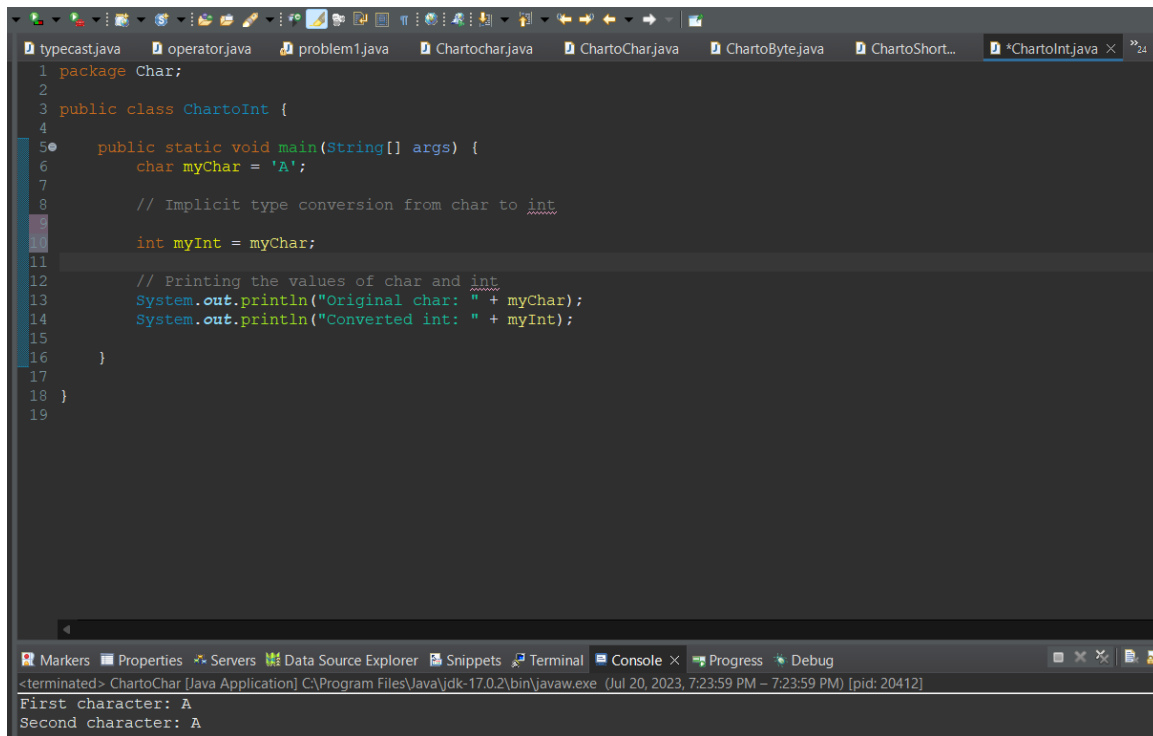
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19
```

<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM - 7:23:59 PM) [pid: 20412]
First character: A
Second character: A

7) Char to Double conversion:

In Java, you can perform an explicit type conversion from **char** to **double**. The **char** data type represents a 16-bit Unicode character, whereas the **double** data type is a 64-bit double-precision floating-point number.

Example:



```
1 package Char;
2
3 public class Chartoint {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
19
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console Progress Debug

<terminated> ChartoChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM – 7:23:59 PM) [pid: 20412]

First character: A
Second character: A

8) Char to Boolean conversion:

In Java, you cannot directly cast a **char** to a **boolean** because they are incompatible data types. The **char** data type represents a 16-bit Unicode character, whereas the **boolean** data type can only hold **true** or **false**.

If you want to convert a **char** value to a **boolean**, you need to define a condition or a rule to determine whether the **char** value should be treated as **true** or **false**.

Result:

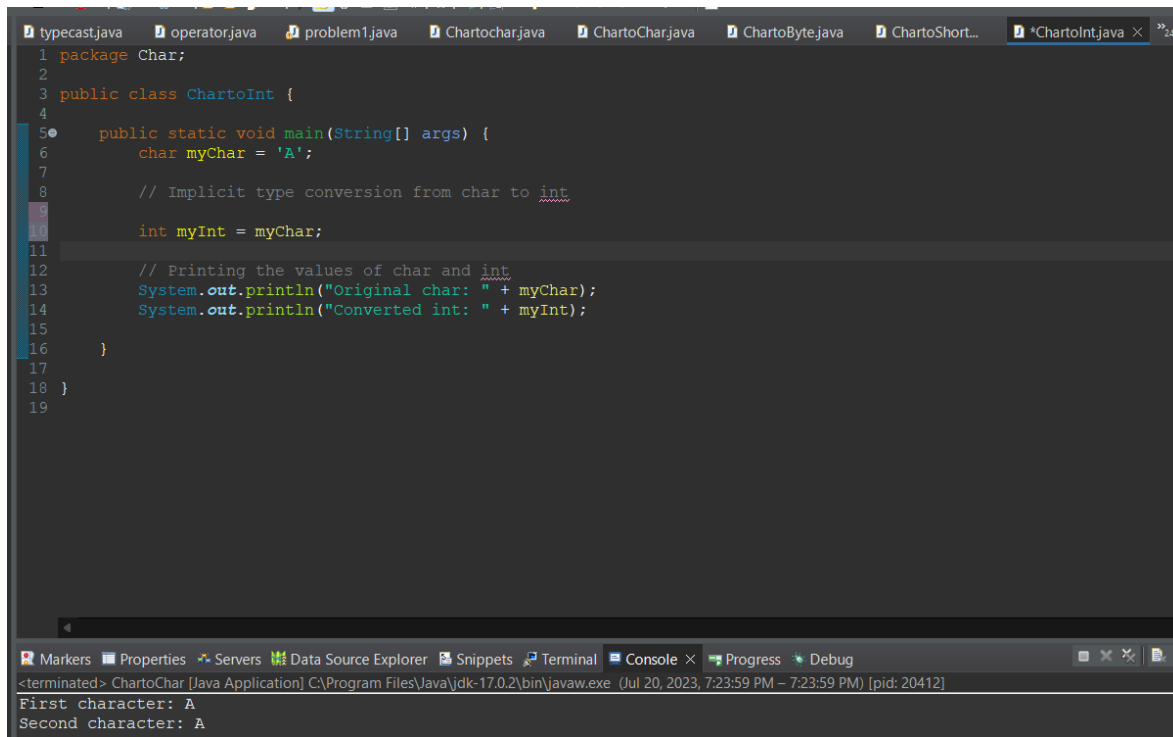
None

Byte Type Conversions:

1) Byte to Char conversion:

In Java, converting a **byte** to a **char** involves explicit type casting because **byte** is an integral type, and **char** is a character data type.

Example:



```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19
```

First character: A
Second character: A

2) Byte to byte conversion:

Converting a **byte** to another **byte** doesn't require any explicit type casting, as it's already the same data type.

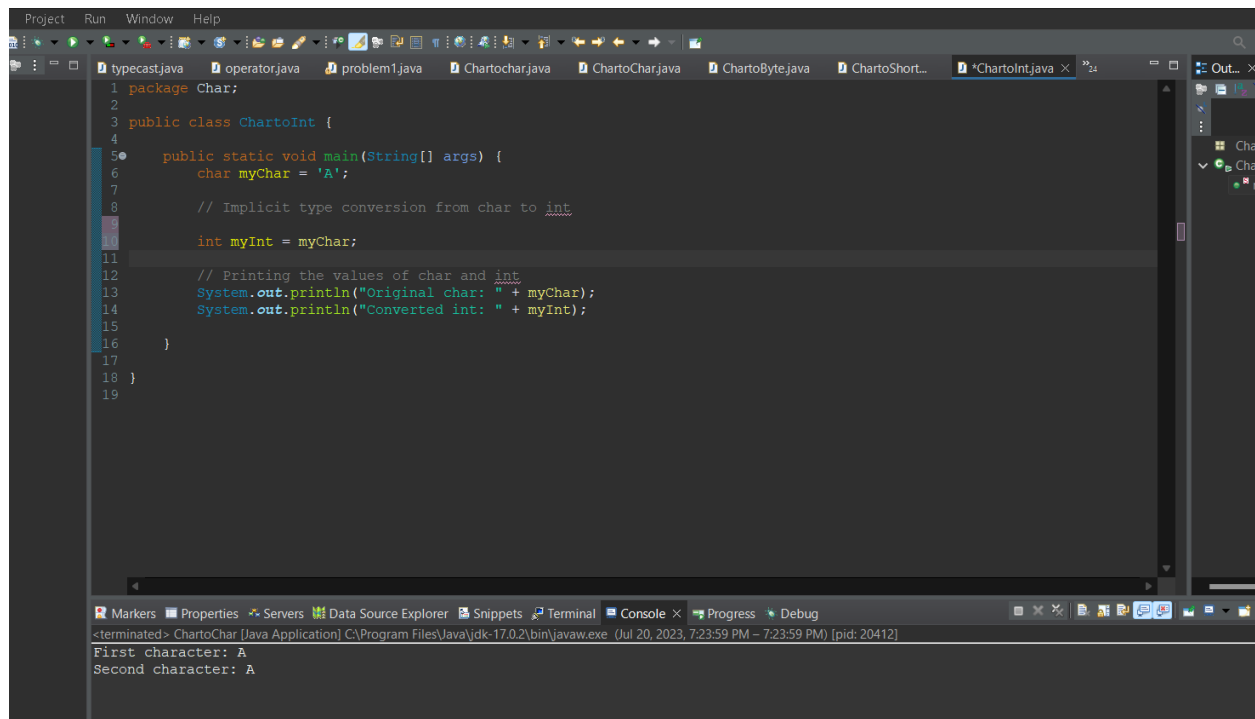
Result:

No conversion required for char to char.

3) Byte to short conversion:

Converting a **byte** to a **short** involves implicit type conversion, as the **short** data type can safely hold the range of values that can be represented by a **byte**.

Example:



```
1 package Char;
2
3 public class ChartoInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17 }
18
19
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console × Progress Debug

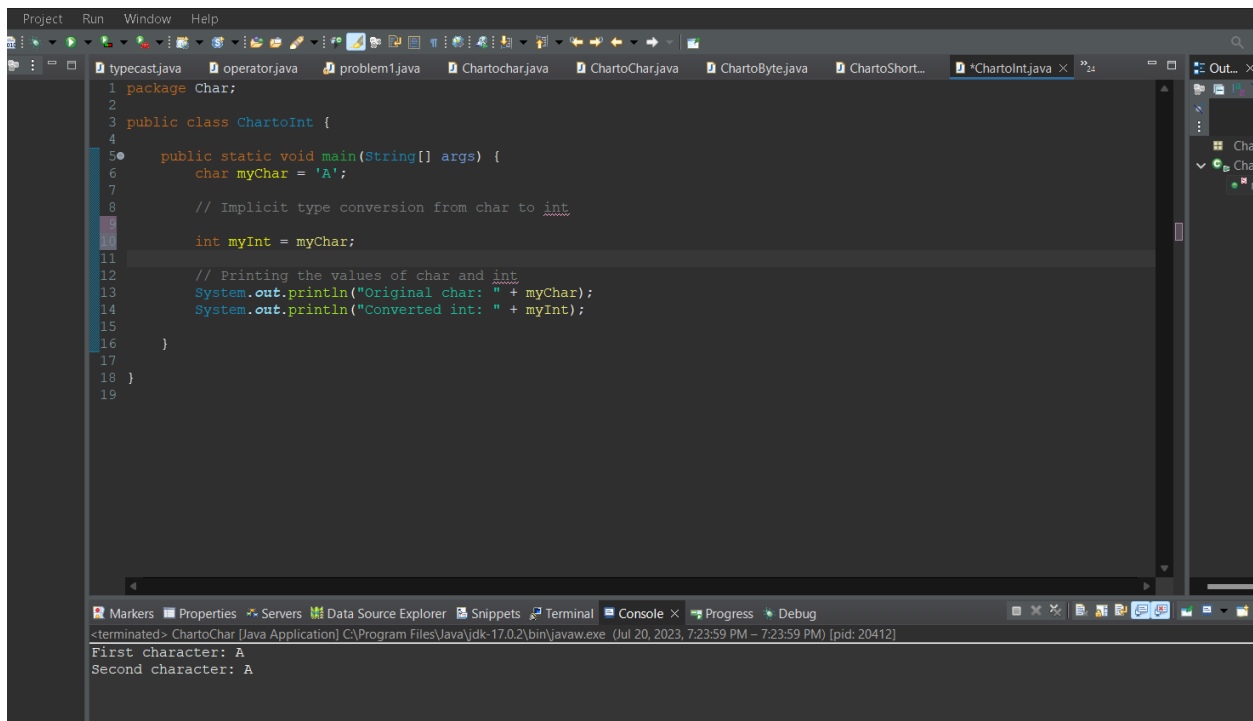
<terminated> ChartoChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]

First character: A
Second character: A

4) Byte to int conversion:

Converting a **byte** to an **int** involves implicit type conversion, as the **int** data type can safely hold the range of values that can be represented by a **byte**.

Example:



```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console Progress Debug

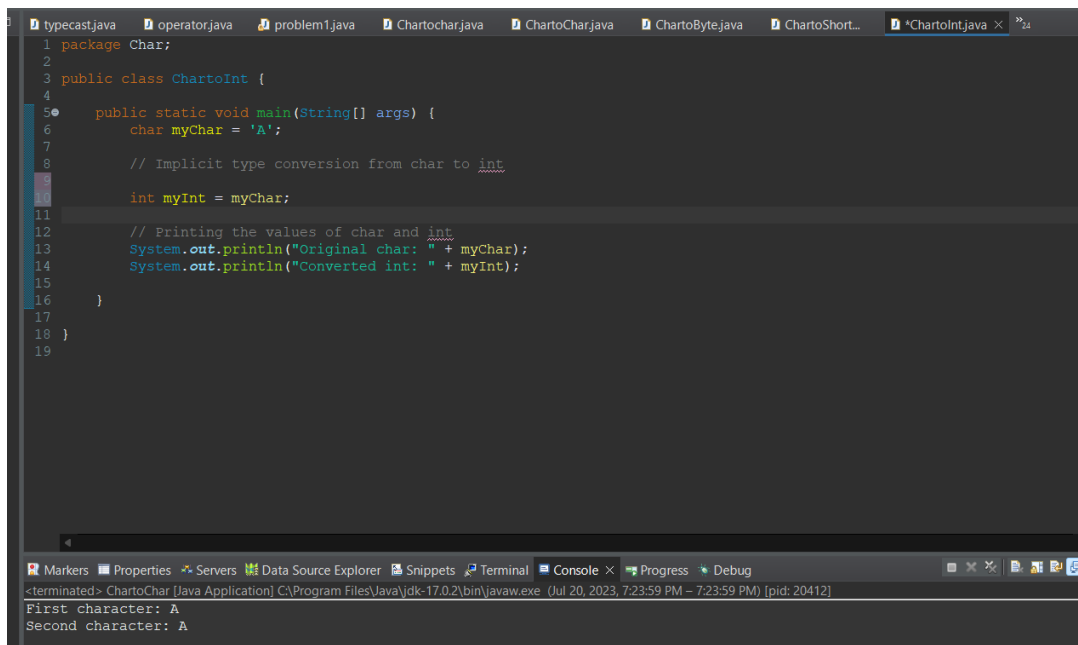
<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]

First character: A
Second character: A

5) Byte to Long conversion:

Converting a **byte** to a **long** involves implicit type conversion, as the **long** data type can safely hold the range of values that can be represented by a **byte**.

Example:



```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console Progress Debug

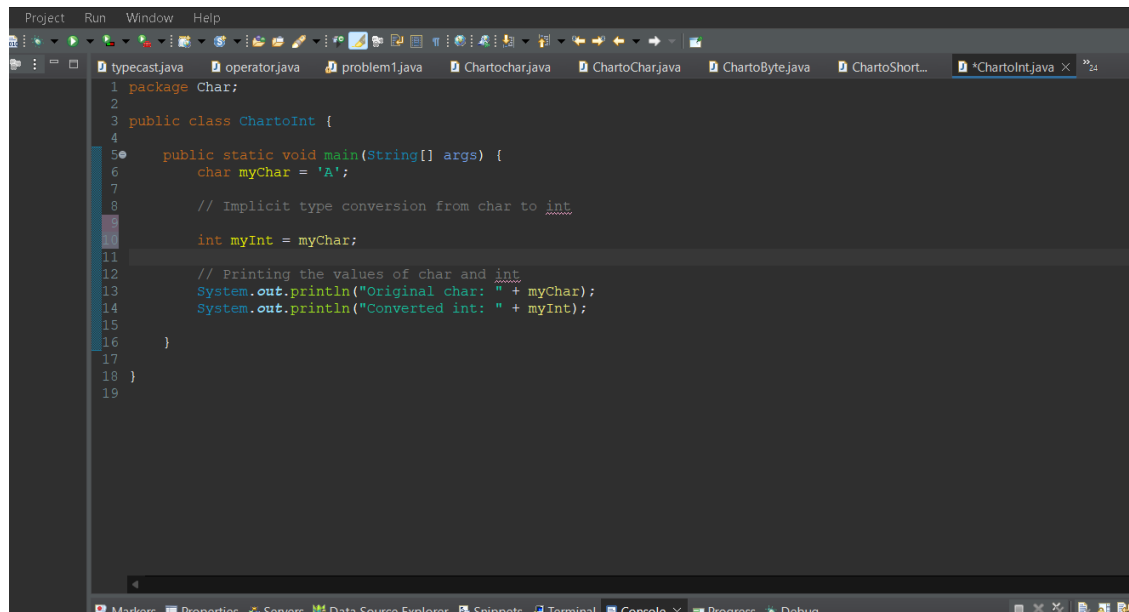
<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]

First character: A
Second character: A

6) Byte to Float Conversion:

Converting a **byte** to a **float** involves implicit type conversion, as the **float** data type can safely hold the range of values that can be represented by a **byte**.

Example:

A screenshot of an IDE window showing a Java file named 'CharToInt.java'. The code defines a class 'CharToInt' with a 'main' method. Inside 'main', a 'char' variable 'myChar' is assigned the value 'A'. Then, an 'int' variable 'myInt' is assigned the value of 'myChar', demonstrating implicit conversion. Finally, the original character and the converted integer are printed to the console.

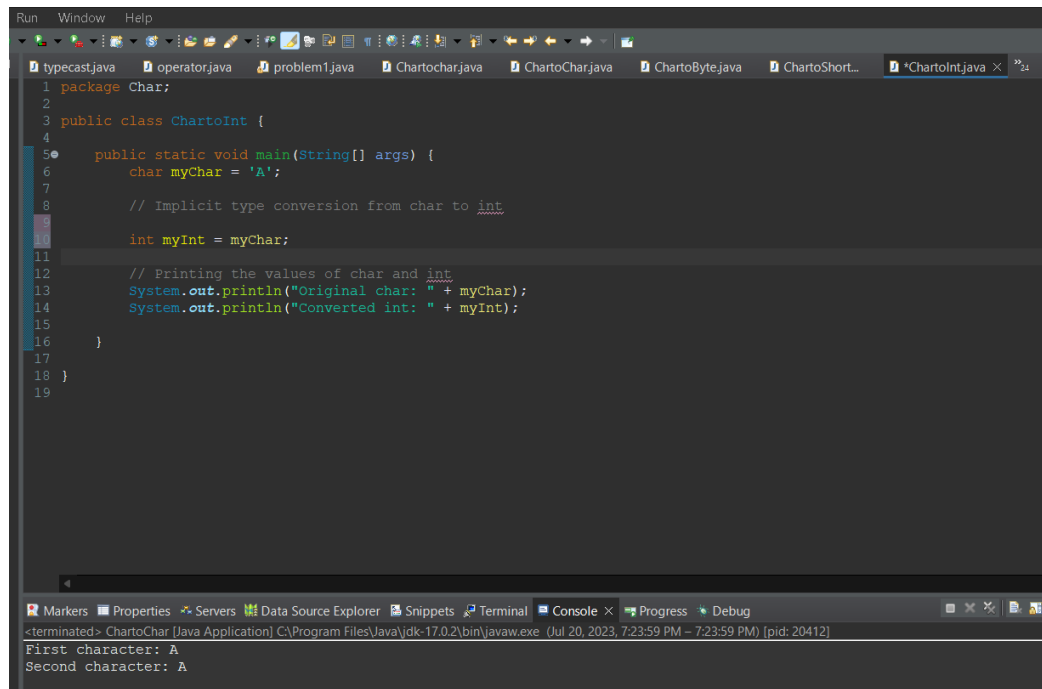
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9         int myInt = myChar;
10
11         // Printing the values of char and int
12         System.out.println("Original char: " + myChar);
13         System.out.println("Converted int: " + myInt);
14     }
15 }
16
17
18
19
```

7) Byte to Double conversion:

Converting a **Byte** wrapper object or a **byte** primitive to a **double** involves implicit type conversion, as the **double** data type can safely hold the range of values that can be represented by a **Byte** or **byte**.

Example:

.



```
Run Window Help
typecast.java operator.java problem1.java CharToChar.java CharToChar.java CharToByte.java CharToShort... CharToInt.java x
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17
18 }
19

Markers Properties Servers Data Source Explorer Snippets Terminal Console x Progress Debug
<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM - 7:23:59 PM) [pid: 20412]
First character: A
Second character: A
```

8)Byte to Boolean conversion:

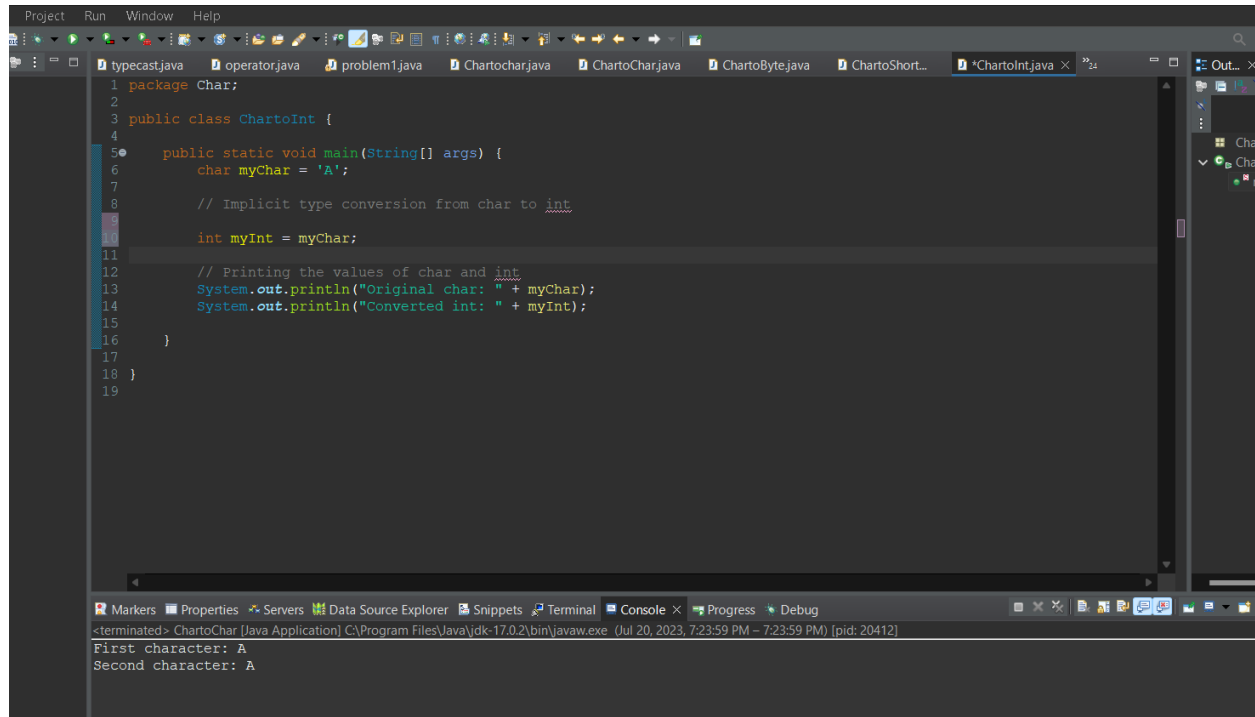
In Java, there is no direct conversion between **byte** and **boolean** data types. Converting a **byte** to a **boolean** cannot be done implicitly or explicitly through type casting, as they are not compatible types.

Result: None

1) Short to Char conversion:

In Java, converting a `short` to a `char` involves explicit type casting because `short` is an integral data type, and `char` is a character data type.

Example:

A screenshot of an IDE window showing a Java file named 'CharToInt.java'. The code defines a class 'CharToInt' with a 'main' method. Inside 'main', a 'char' variable 'myChar' is assigned the value 'A'. Then, an 'int' variable 'myInt' is assigned the value of 'myChar', demonstrating implicit conversion from char to int. Finally, two print statements are executed: 'System.out.println("Original char: " + myChar);' and 'System.out.println("Converted int: " + myInt);'. The IDE's console at the bottom shows the output: 'First character: A' and 'Second character: A'.

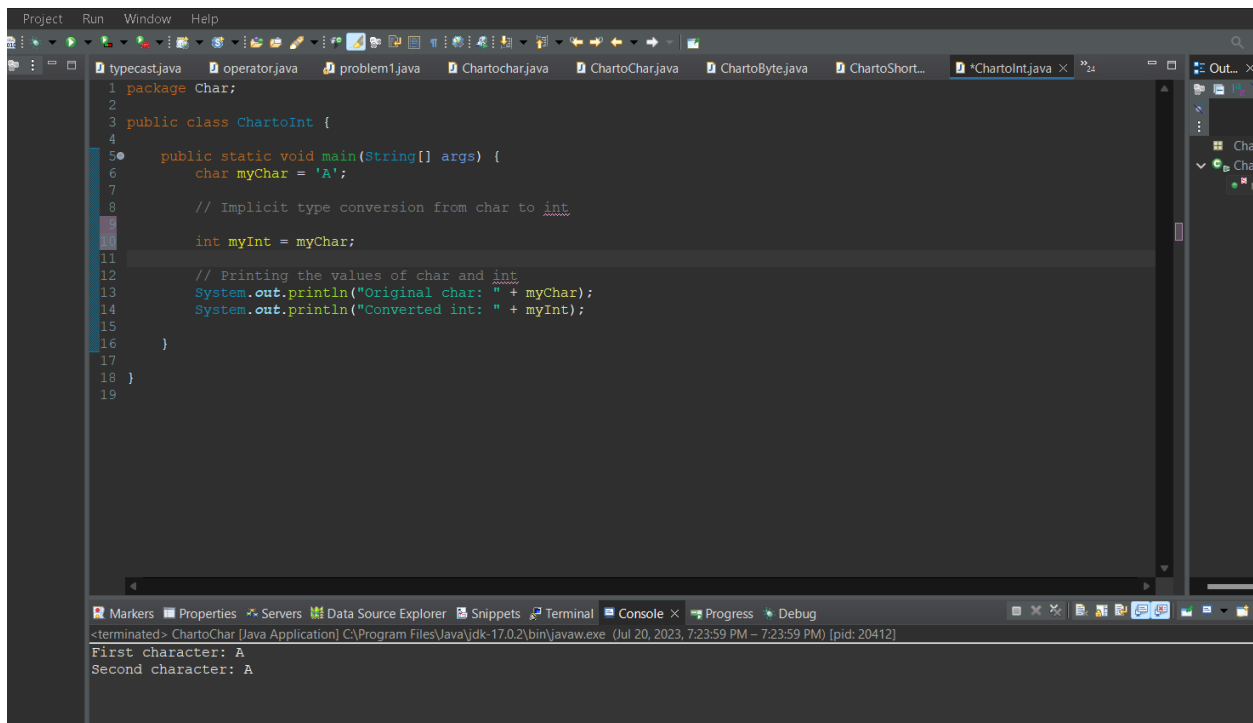
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
19
```

<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]
First character: A
Second character: A

2) Short to Byte conversion:

In Java, converting a `short` to a `byte` involves explicit type casting because `short` is a 16-bit integral data type, and `byte` is an 8-bit integral data type. Therefore, a narrowing conversion is required, and you need to explicitly cast the `short` value to a `byte`.

Example:

A screenshot of an IDE window showing a Java file named 'CharToInt.java'. The code defines a package 'Char' and a class 'CharToInt' with a 'main' method. Inside 'main', a 'char' variable 'myChar' is assigned the value 'A'. A comment indicates an implicit conversion from 'char' to 'int'. An 'int' variable 'myInt' is then assigned the value of 'myChar'. Finally, two lines of code use 'System.out.println' to print the original character and the converted integer. The IDE's output console at the bottom shows the program's execution, displaying 'First character: A' and 'Second character: A'.

```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17 }
18
19
```

First character: A
Second character: A

3) Short to Short conversion:

Converting a **short** to another **short** doesn't require any explicit type casting, as it's already the same data type. It is essentially an identity conversion, meaning the value remains the same.

Result: None

4) Short to int conversion:

Converting a **short** to an **int** involves implicit type conversion, as the **int** data type can safely hold the range of values that can be represented by a **short**.

Example:

The screenshot shows an IDE with a Java file named `CharToInt.java`. The code is as follows:

```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
```

The IDE's console window at the bottom shows the output of the program:

```
<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]
First character: A
Second character: A
```

5) Short to Long conversion:

In Java, converting a **short** to an **int** or a **long** involves implicit type conversion, as both **int** and **long** data types can safely hold the range of values that can be represented by a **short**.

Example:

The screenshot shows an IDE with a Java file named `CharToInt.java`. The code is as follows:

```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
```

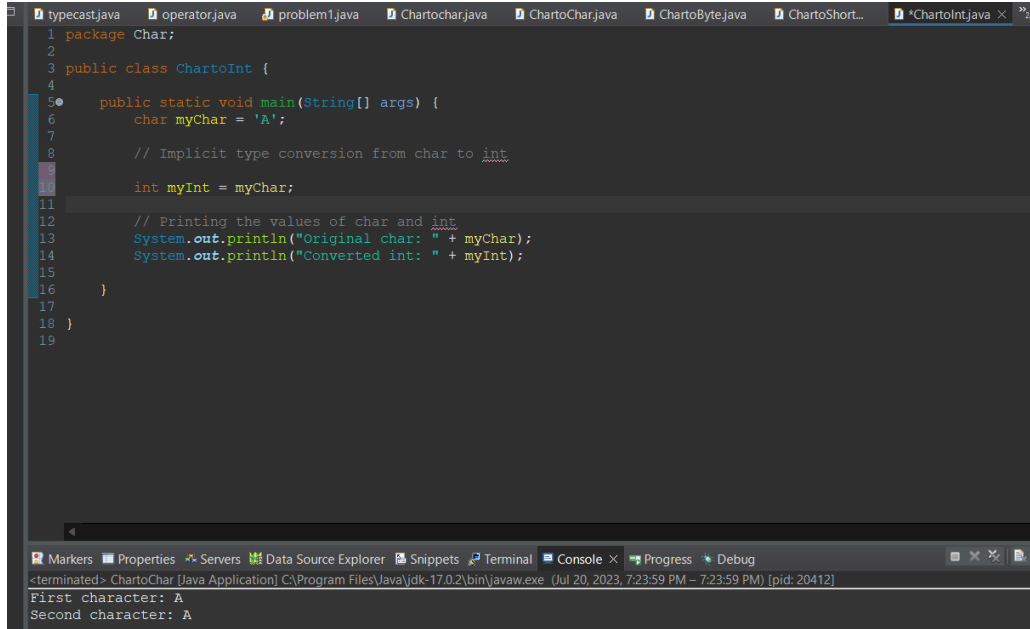
The IDE's console window at the bottom shows the output of the program:

```
<terminated> CharToChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]
First character: A
Second character: A
```

6) Short to float conversion:

In Java, converting a **short** to a **float** involves implicit type conversion, as the **float** data type can safely hold the range of values that can be represented by a **short**.

Example:



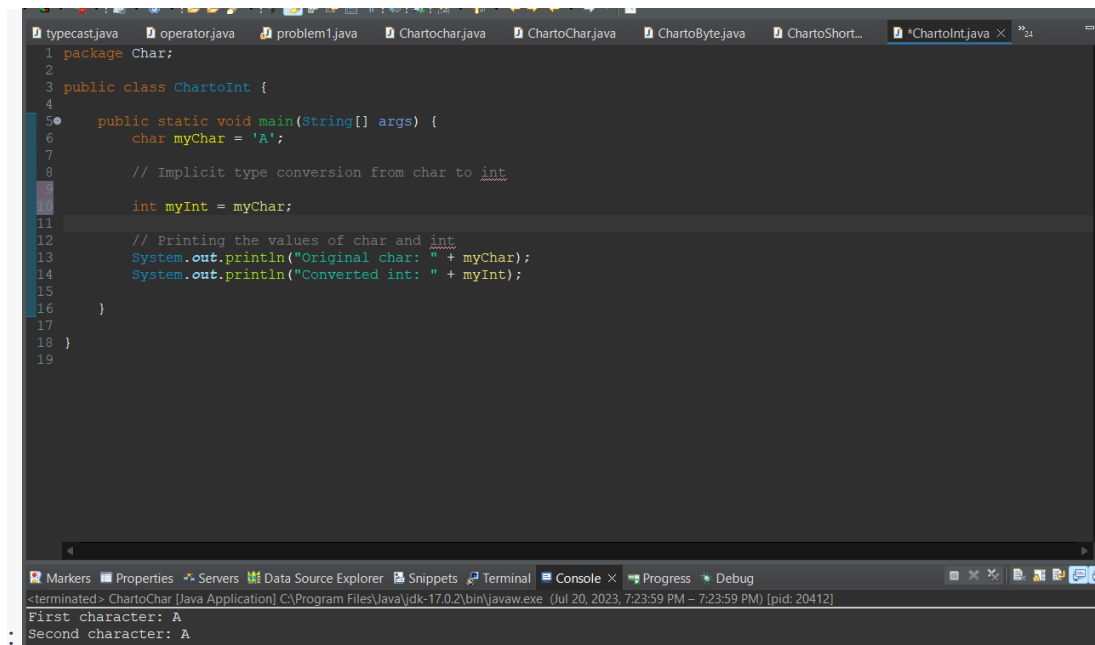
```
1 package Char;
2
3 public class ChartoInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
19
```

First character: A
Second character: A

7) Short to Double conversion:

In Java, converting a **short** to a **double** involves implicit type conversion, as the **double** data type can safely hold the range of values that can be represented by a **short**.

Example:



```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9         int myInt = myChar;
10
11         // Printing the values of char and int
12         System.out.println("Original char: " + myChar);
13         System.out.println("Converted int: " + myInt);
14     }
15 }
16
17
18
19
```

First character: A
Second character: A

8) Short to boolean conversion:

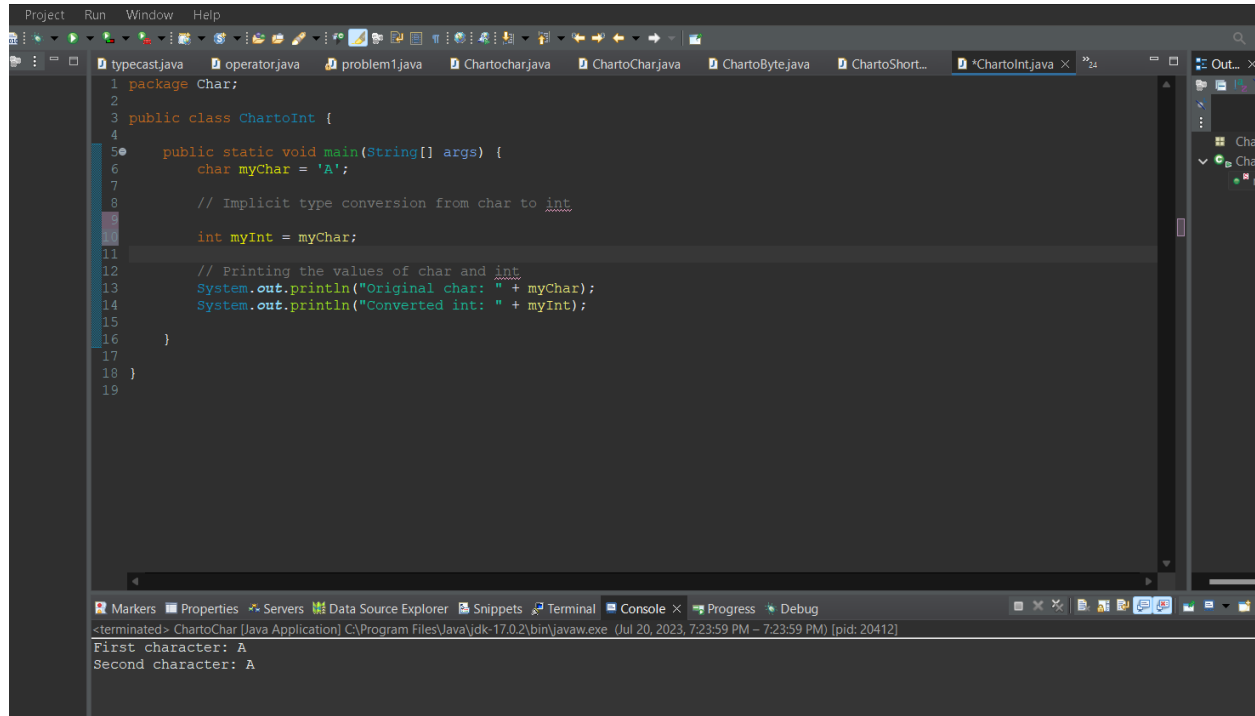
In Java, there is no direct conversion between **short** and **boolean** data types. Converting a **short** to a **boolean** cannot be done implicitly or explicitly through type casting, as they are not compatible types.

Result: None

1) Int to Char conversion in java:

In Java, converting an **int** to a **char** involves explicit type casting because **int** is an integral data type, and **char** is a character data type.

Example:



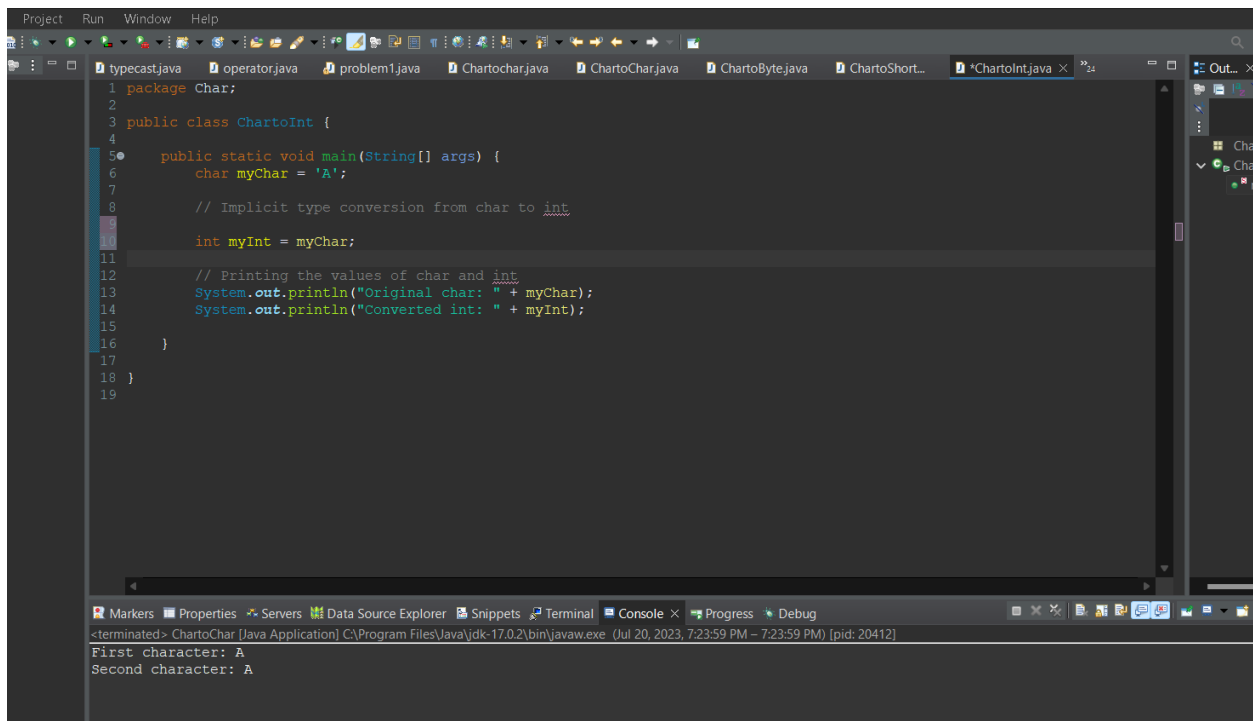
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
19
```

First character: A
Second character: A

2) Int to Byte conversion:

In Java, converting an **int** to a **byte** involves explicit type casting because **int** is a 32-bit integral data type, and **byte** is an 8-bit integral data type. Therefore, a narrowing conversion is required, and you need to explicitly cast the **int** value to a **byte**.

Example:



```
1 package Char;
2
3 public class ChartoInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17 }
18
19
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console Progress Debug

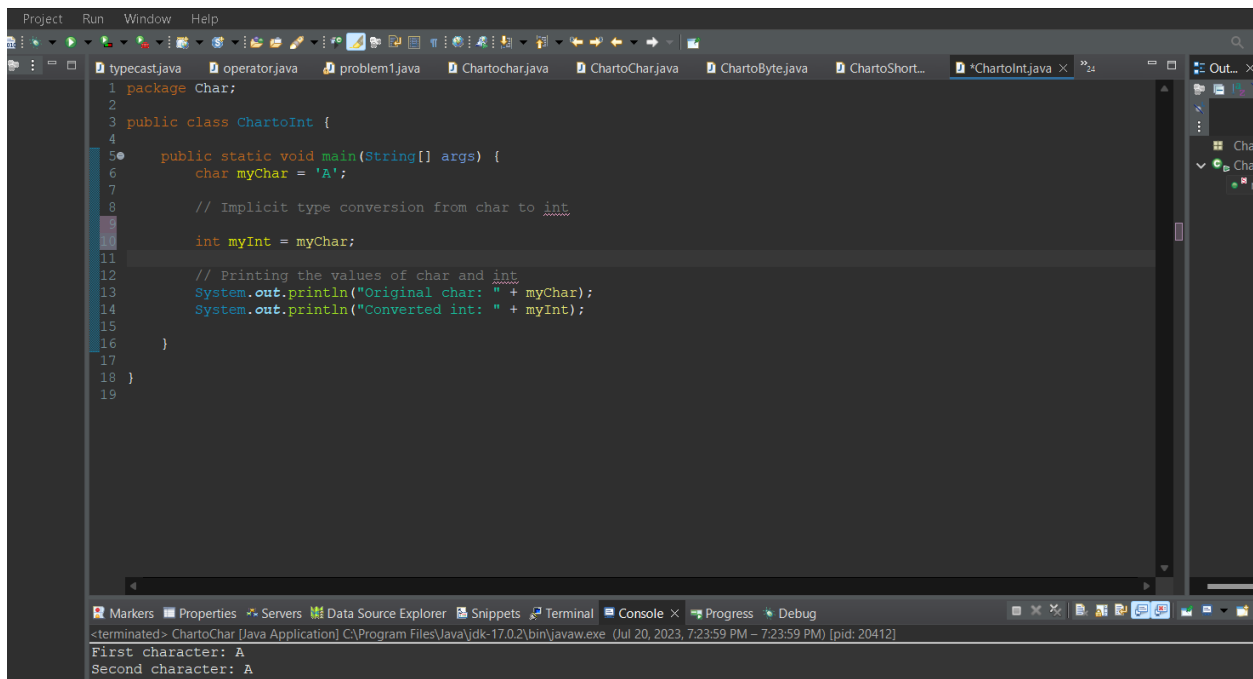
<terminated> ChartoChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]

First character: A
Second character: A

3) Int to Short conversion:

In Java, converting an **int** to a **short** involves explicit type casting because **int** is a 32-bit integral data type, and **short** is a 16-bit integral data type. Therefore, a narrowing conversion is required, and you need to explicitly cast the **int** value to a **short**.

Example:



```
1 package Char;
2
3 public class ChartoInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17 }
18
19
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console Progress Debug

<terminated> ChartoChar [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (Jul 20, 2023, 7:23:59 PM) [pid: 20412]

First character: A
Second character: A

4) Int to Int conversion:

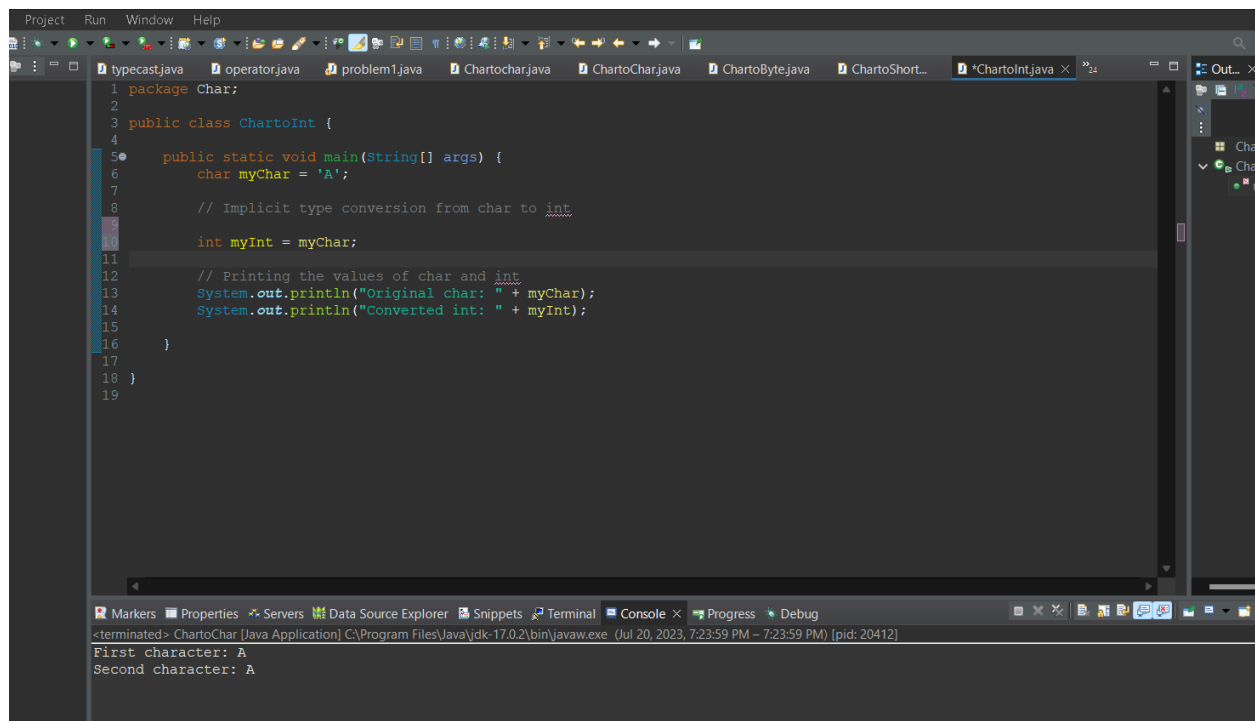
It seems there might be a typo in your question. If you meant "int to int" conversion in Java, then that would be a simple identity conversion where no explicit type casting is required, as `int` is already an `int`.

Result: Not required

5) Int to Long conversion:

In Java, converting an `int` to a `long` involves implicit type conversion, as the `long` data type can safely hold the range of values that can be represented by an `int`.

Example:



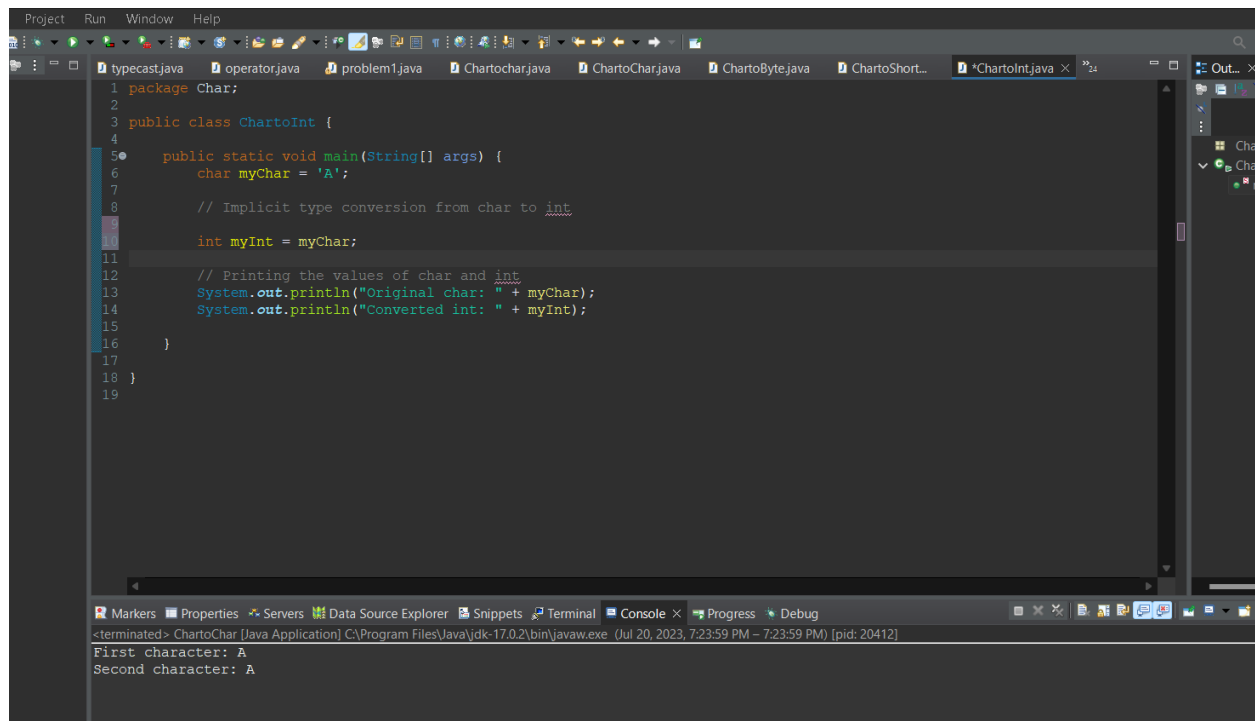
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15
16    }
17 }
18
19
```

First character: A
Second character: A

6) Int to Float conversion:

In Java, converting an `int` to a `float` involves implicit type conversion, as the `float` data type can safely hold the range of values that can be represented by an `int`.

Example:



The screenshot shows an IDE with a Java file named `CharToInt.java`. The code defines a package `Char` and a class `CharToInt` with a `main` method. Inside `main`, a `char myChar = 'A';` is declared. A comment indicates an implicit conversion from `char` to `int`. The variable `myInt` is assigned the value of `myChar`. Two `println` statements are used to print the original character and the converted integer value. The IDE's output console at the bottom shows the execution results: "First character: A" and "Second character: A".

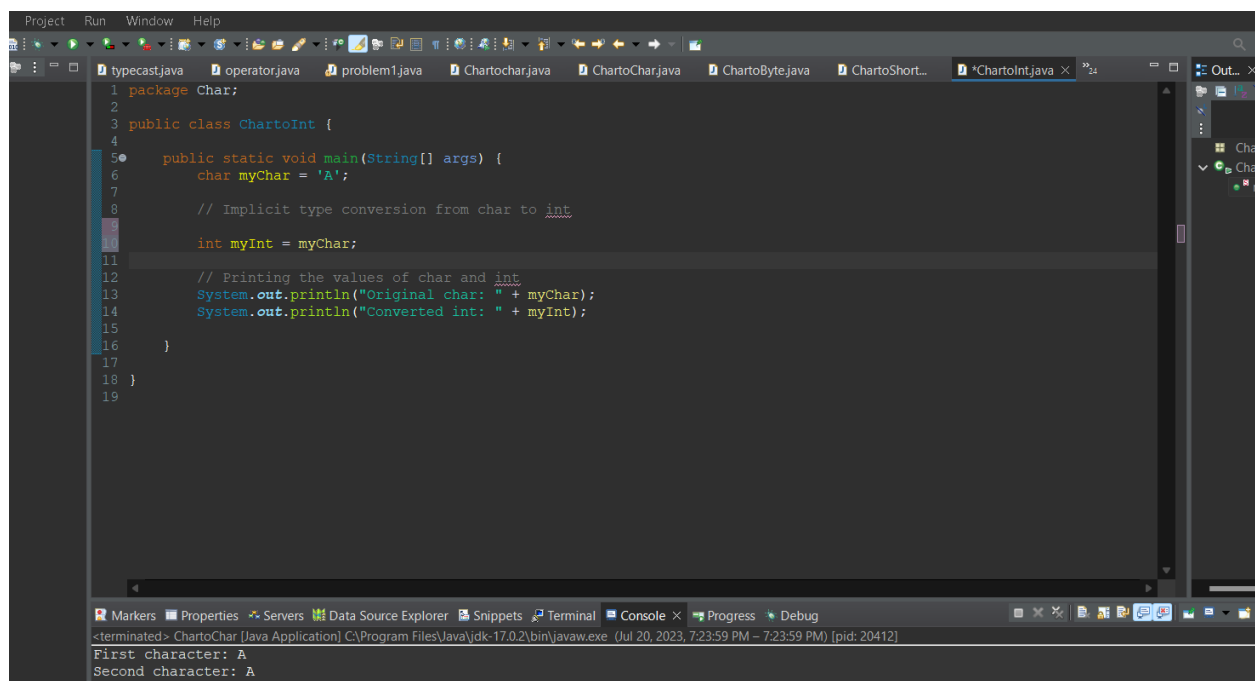
```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
19
```

First character: A
Second character: A

7)Int to Double conversion:

In Java, converting an **int** to a **double** involves implicit type conversion, as the **double** data type can safely hold the range of values that can be represented by an **int**.

Example:



This screenshot is identical to the one above, showing the same IDE environment and Java code for char to int conversion. The code in `CharToInt.java` remains the same, and the output console still displays "First character: A" and "Second character: A".

```
1 package Char;
2
3 public class CharToInt {
4
5     public static void main(String[] args) {
6         char myChar = 'A';
7
8         // Implicit type conversion from char to int
9
10        int myInt = myChar;
11
12        // Printing the values of char and int
13        System.out.println("Original char: " + myChar);
14        System.out.println("Converted int: " + myInt);
15    }
16 }
17
18 }
19
```

First character: A
Second character: A

8) Int to Boolean conversion:

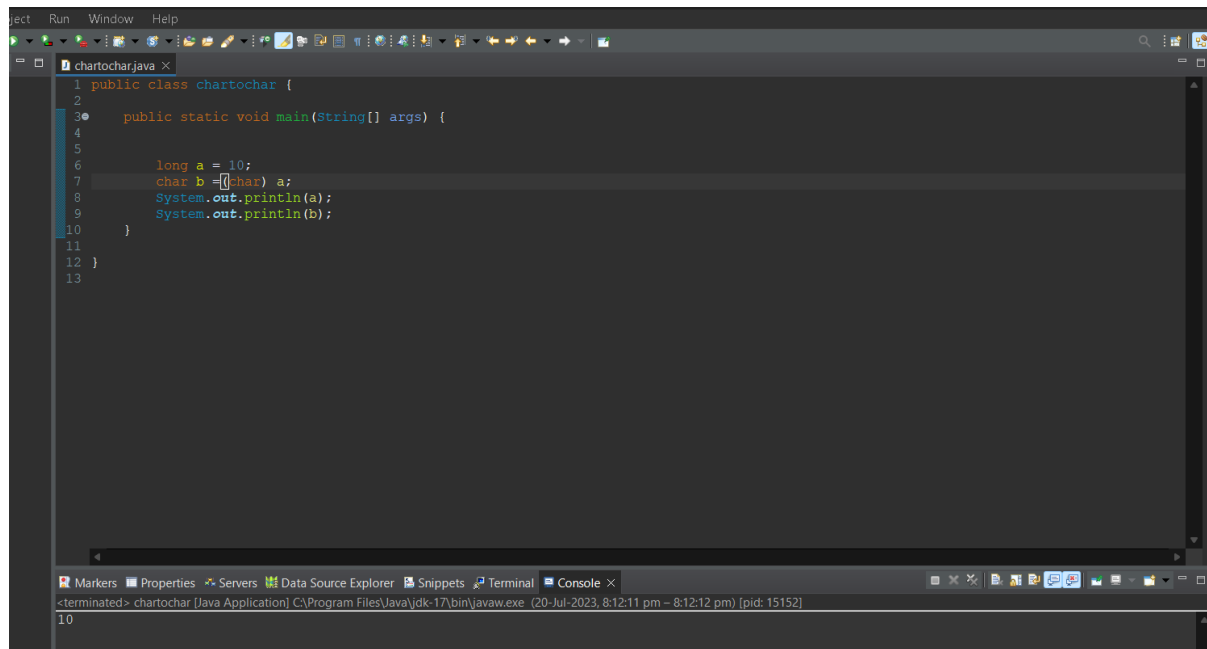
In Java, there is no direct conversion between **int** and **boolean** data types. Converting an **int** to a **boolean** cannot be done implicitly or explicitly through type casting, as they are not compatible types.

Result: No conversion

1) Long to char explicit :

In Java, you can convert a **long** data type to a **char** data type using type casting or by converting the **long** to a **String** first and then getting the **char** from the **String**. Let's see both methods:

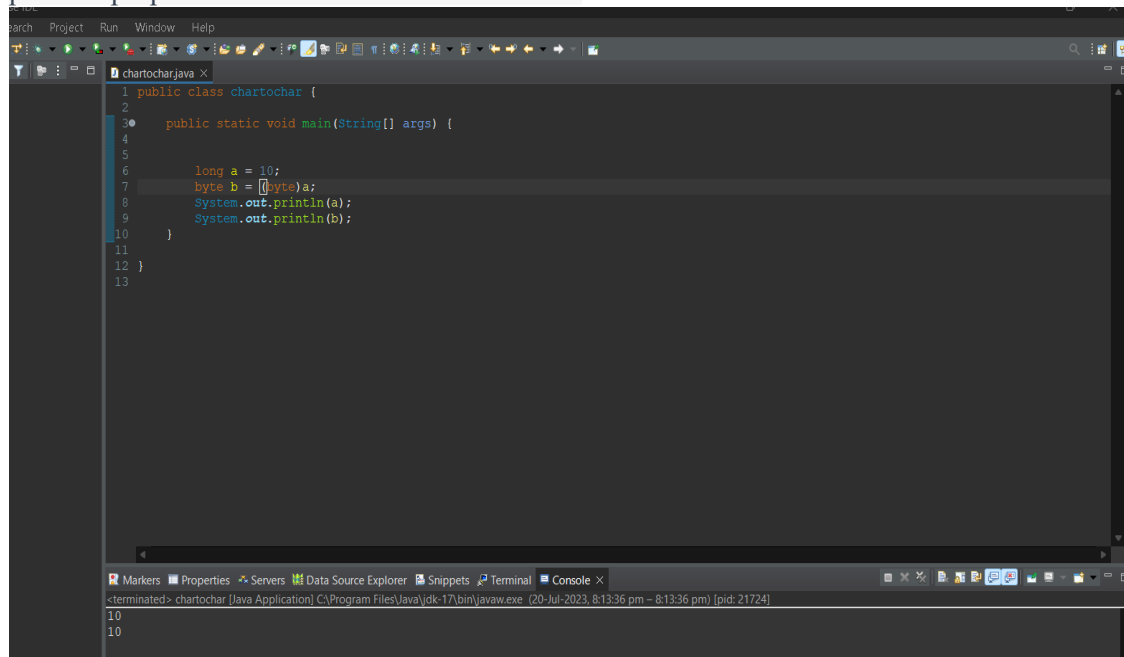
Example:

A screenshot of an IDE window titled 'chartochar.java'. The code defines a public class 'chartochar' with a main method. Inside the main method, a long variable 'a' is assigned the value 10. Then, a char variable 'b' is assigned the value of (char) a. Finally, System.out.println(a) and System.out.println(b) are called. The IDE interface includes a toolbar at the top, a sidebar on the left, and a bottom panel with tabs for 'Markers', 'Properties', 'Servers', 'Data Source Explorer', 'Snippets', 'Terminal', and 'Console'. The console shows the output of the program, which is '10'.

2) Long to byte explicit:

In Java, you can convert a **long** data type to a **byte** data type using type casting. However, be aware that a **long** value may not always fit into a **byte**, as **long** can store much larger values than **byte**. If the **long** value is outside the valid range of a **byte**, data loss may occur. To avoid potential data loss, you should

perform proper validation before the conversion.

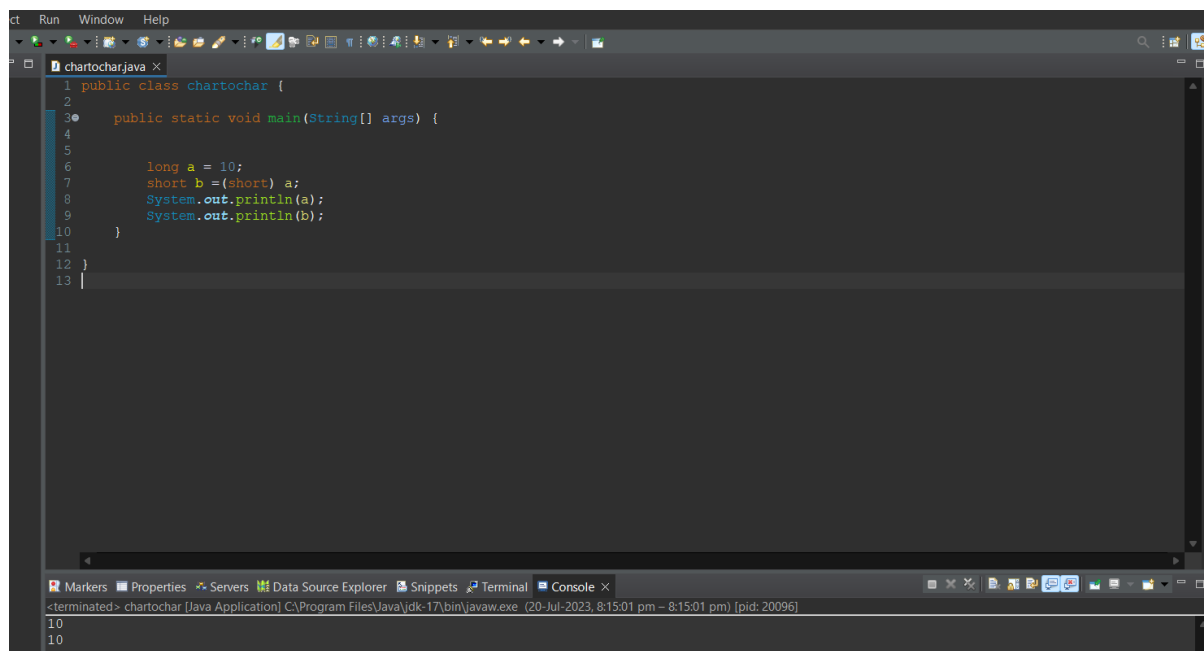


```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         long a = 10;
6         byte b = (byte)a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The screenshot shows an IDE window titled 'chartochar.java'. The code defines a class with a main method. Inside the main method, a long variable 'a' is assigned the value 10. This value is then cast to a byte variable 'b' using the expression '(byte)a'. The program then prints the values of 'a' and 'b' using 'System.out.println()'. The console output at the bottom shows '10' on two separate lines, indicating that both 'a' and 'b' contain the value 10.

3) Long to short explicit:

In Java, you can convert a **long** data type to a **short** data type using type casting. However, similar to the previous example, you need to be cautious about potential data loss since a **long** can store larger values than a **short**. You should ensure that the **long** value is within the valid range of a **short** before performing the conversion.



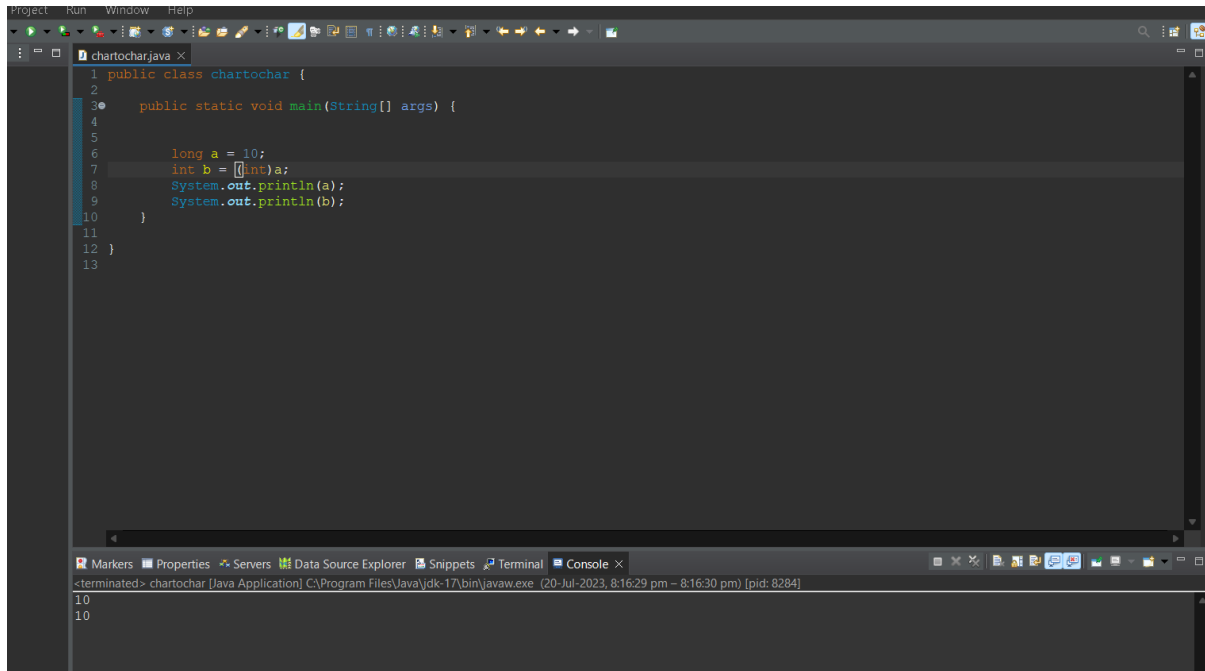
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         long a = 10;
6         short b = (short) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The screenshot shows an IDE window titled 'chartochar.java'. The code defines a class with a main method. Inside the main method, a long variable 'a' is assigned the value 10. This value is then cast to a short variable 'b' using the expression '(short) a'. The program then prints the values of 'a' and 'b' using 'System.out.println()'. The console output at the bottom shows '10' on two separate lines, indicating that both 'a' and 'b' contain the value 10.

4) Long to int explicit:

In Java, you can convert a **long** data type to an **int** data type using type casting. However, similar to previous examples, you should be careful about potential data loss since a **long** can store larger values than an **int**. You need to ensure that the **long** value is within the valid range of an **int** before performing the conversion.

Example:



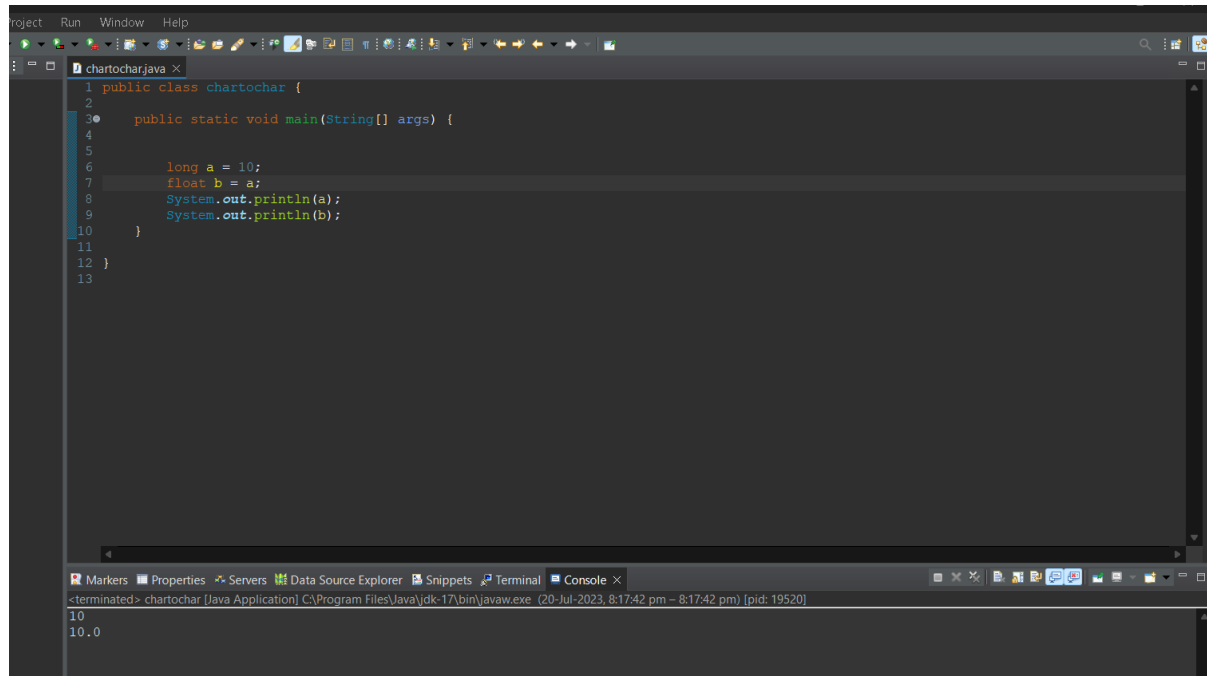
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         long a = 10;
6         int b = (int)a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The screenshot shows an IDE window titled 'chartochar.java'. The code defines a class 'chartochar' with a 'main' method. Inside 'main', a 'long' variable 'a' is assigned the value 10. Then, 'a' is cast to an 'int' and assigned to 'b'. Finally, both 'a' and 'b' are printed to the console. The bottom of the screenshot shows a terminal window with the output of the program, which is '10' on two separate lines, corresponding to the two print statements in the code.

5) Long to float implicit :

In Java, you can convert a **long** data type to a **float** data type using type casting. However, you should be aware that type casting from **long** to **float** may result in a loss of precision since **float** is a single-precision 32-bit floating-point data type, whereas **long** is a 64-bit integer data type.

Example:



```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         long a = 10;
6         float b = a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

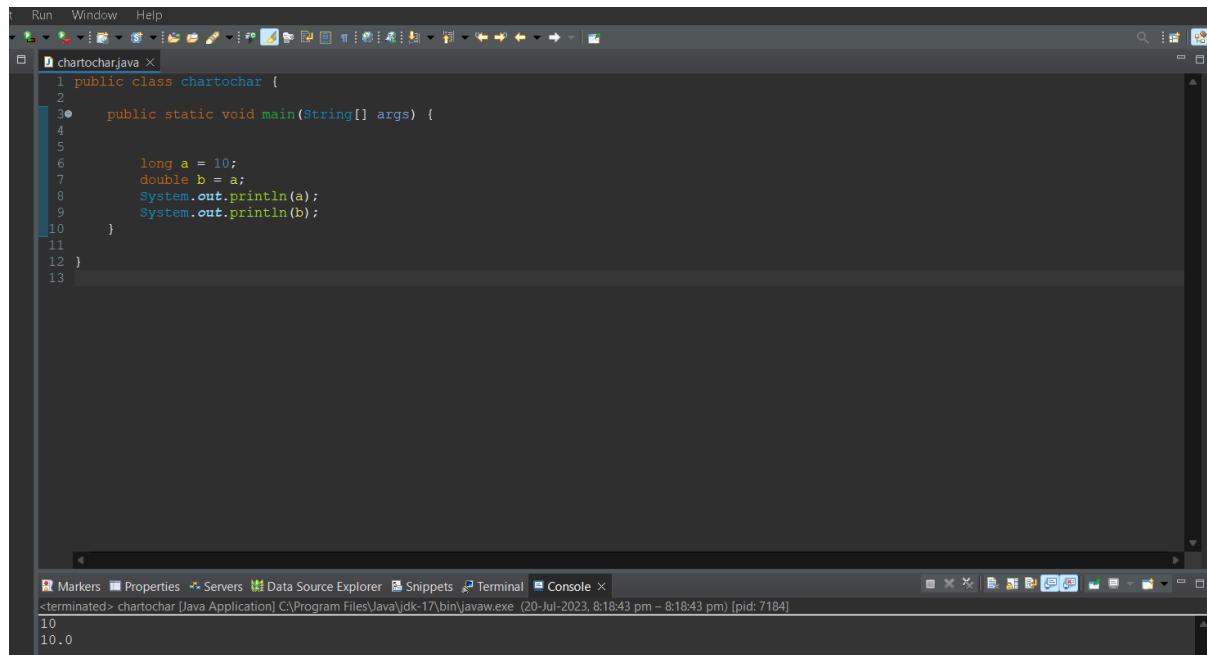
Console output:

```
<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 8:17:42 pm ~ 8:17:42 pm) [pid: 19520]
10
10.0
```

6) Long to double implicit:

In Java, you can convert a **long** data type to a **double** data type using type casting. Converting **long** to **double** does not result in any loss of precision, as **double** is a double-precision 64-bit floating-point data type that can accurately represent a wider range of values, including integers and fractions.

Example:



```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         long a = 10;
6         double b = a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

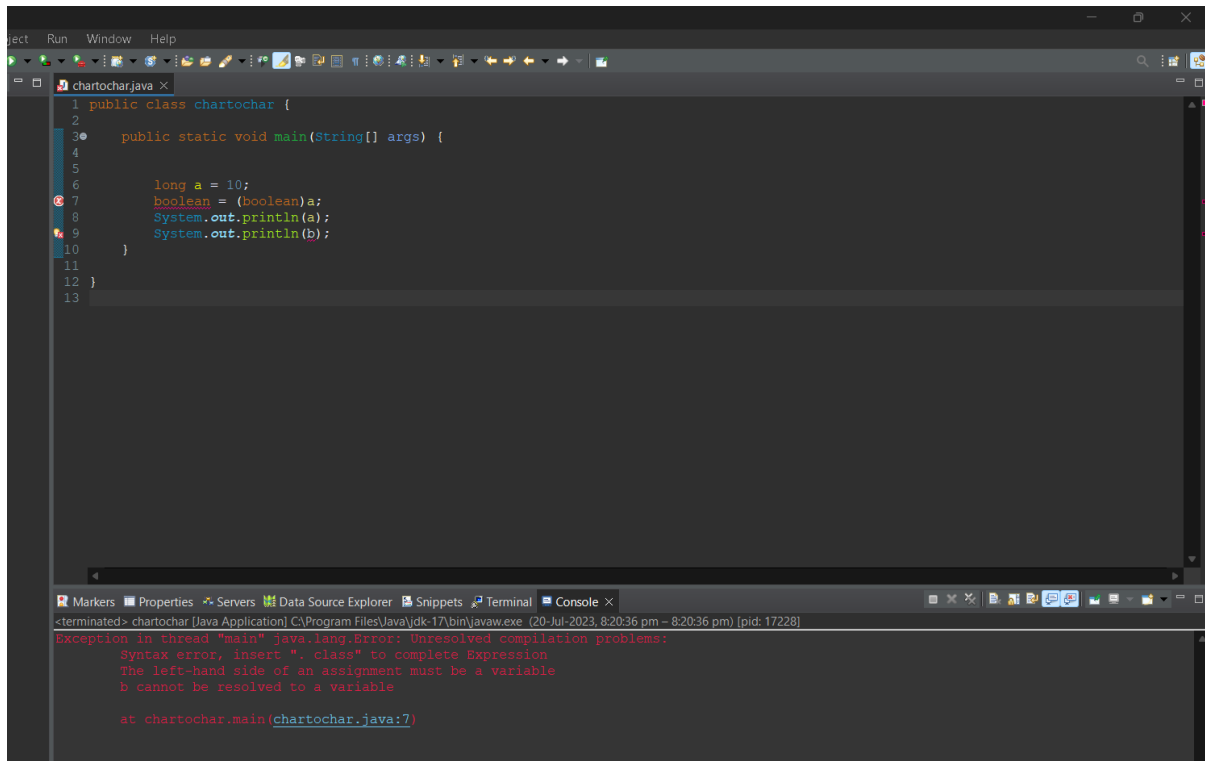
Console output:

```
<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 8:18:43 pm ~ 8:18:43 pm) [pid: 7184]
10
10.0
```

7) Long to Boolean:

In Java, you cannot directly convert a **long** data type to a **boolean** data type using type casting, as there is no direct numeric representation of **true** or **false** in a **long**.

If you want to convert a **long** to a **boolean**, you'll need to define a condition based on your specific use case. For example, you can set a rule such that any non-zero value of **long** will be considered **true**, and **false** otherwise.

A screenshot of an IDE window titled 'chartochar.java'. The code is as follows:

```
1 public class chartochar {  
2  
3     public static void main(String[] args) {  
4  
5  
6         long a = 10;  
7         boolean = (boolean)a;  
8         System.out.println(a);  
9         System.out.println(b);  
10    }  
11  
12 }  
13
```

Red squiggly lines indicate errors on lines 7 and 9. The bottom console shows the following error message:

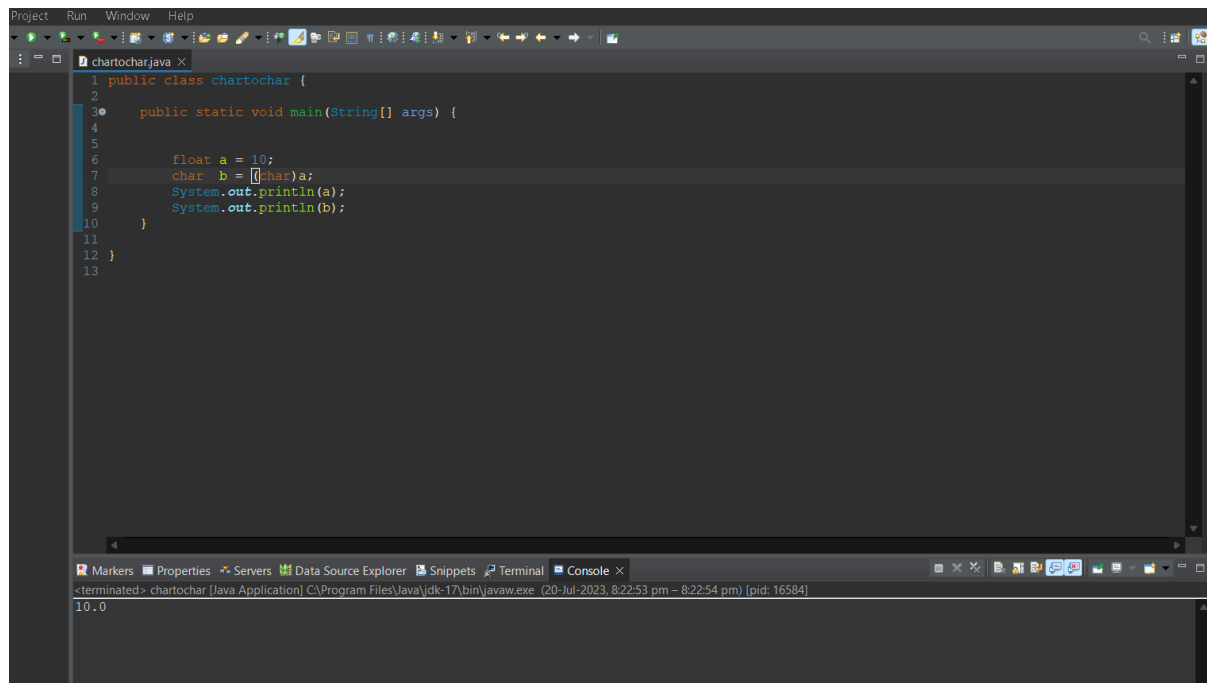
```
<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 8:20:36 pm - 8:20:36 pm) [pid: 17228]  
Exception in thread "main" java.lang.Error: Unresolved compilation problems:  
Syntax error, insert ". class" to complete Expression  
The left-hand side of an assignment must be a variable  
b cannot be resolved to a variable  
  
at chartochar.main(chartochar.java:7)
```

1) Float to char explicit:

In Java, you cannot directly convert a **float** data type to a **char** data type using type casting, as there is no direct numeric representation of characters in a **float**.

To convert a **float** to a **char**, you can convert the **float** to an integer or a **String** first and then extract the character representation from the integer or the first character of the **String**.

Example:



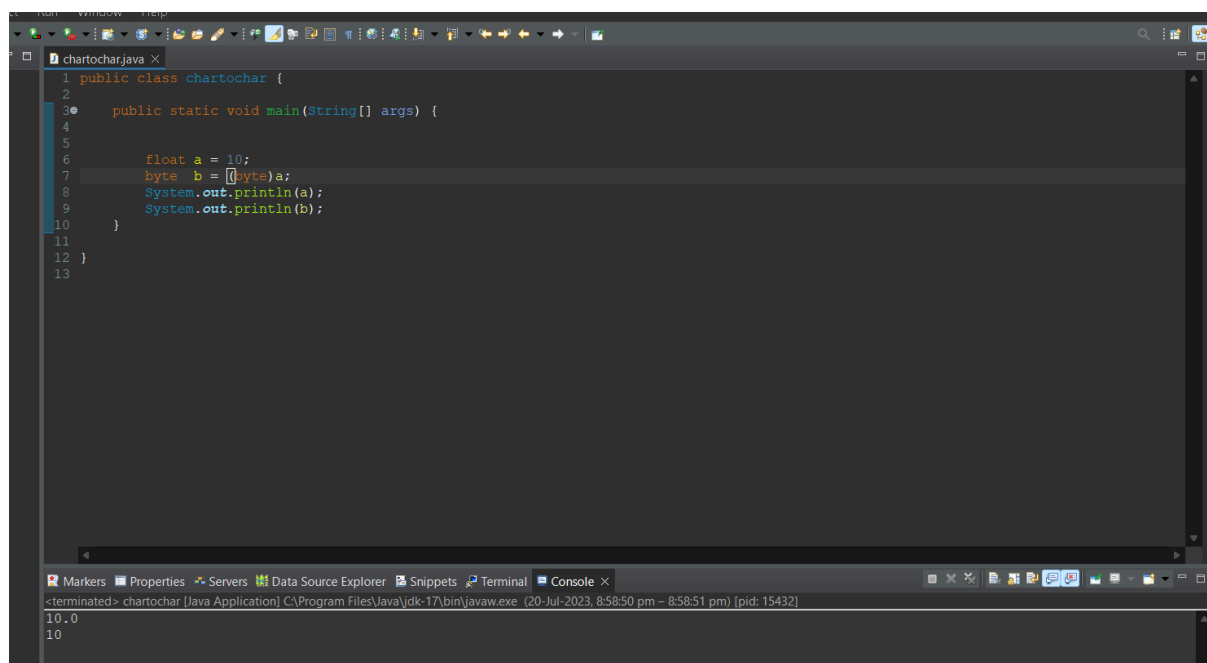
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10;
6         char b = (char)a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The console output shows the value 10.0 on the first line and 10 on the second line, indicating that the float value 10.0 was successfully cast to the char value 10.

2) Float to byte explicit:

In Java, you can convert a **float** data type to a **byte** data type using type casting. However, be aware that converting a **float** to a **byte** may result in a loss of precision, as **byte** is an 8-bit signed integer data type, and **float** is a 32-bit floating-point data type.

Example:



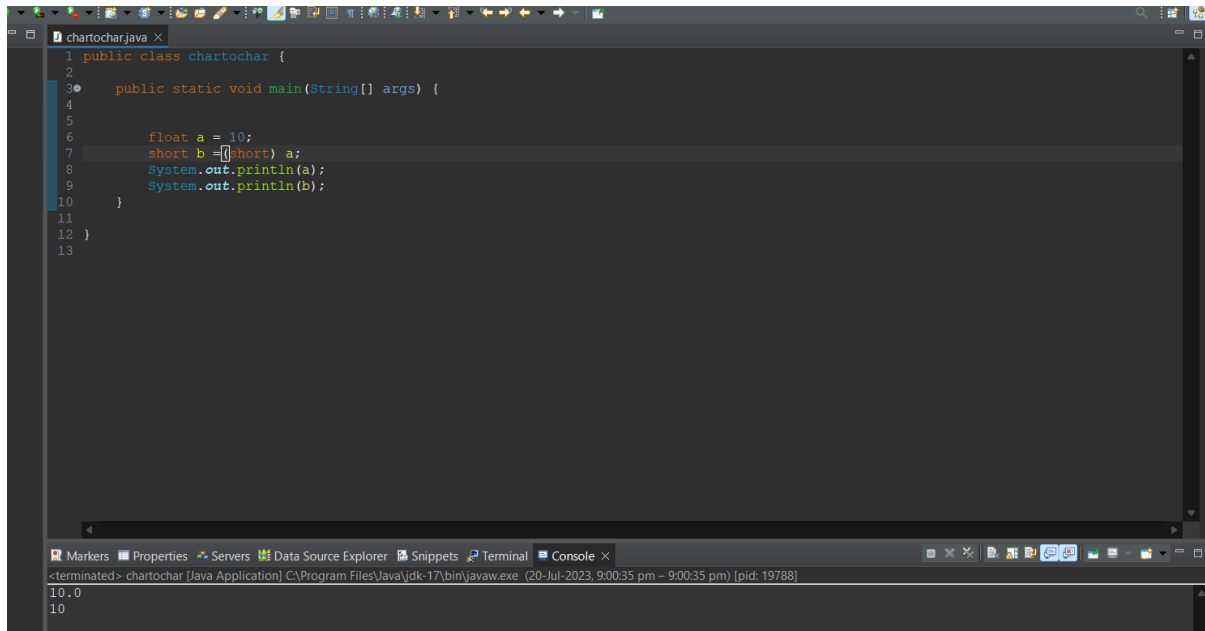
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10;
6         byte b = (byte)a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The console output shows the value 10.0 on the first line and 10 on the second line, indicating that the float value 10.0 was successfully cast to the byte value 10.

3) Float to short explicit:

In Java, you can convert a **float** data type to a **short** data type using type casting. However, similar to previous examples, be aware that converting a **float** to a **short** may result in a loss of precision, as **short** is a 16-bit signed integer data type, and **float** is a 32-bit floating-point data type.

Example:

A screenshot of an IDE window titled 'chartochar.java'. The code defines a public class 'chartochar' with a main method. Inside the main method, a float variable 'a' is assigned the value 10.0. This float is then explicitly cast to a short variable 'b' using the syntax '(short) a'. The program then prints the values of 'a' and 'b' using 'System.out.println'. The output console at the bottom shows the values '10.0' and '10', confirming the successful conversion and truncation of the decimal part.

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10;
6         short b = (short) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

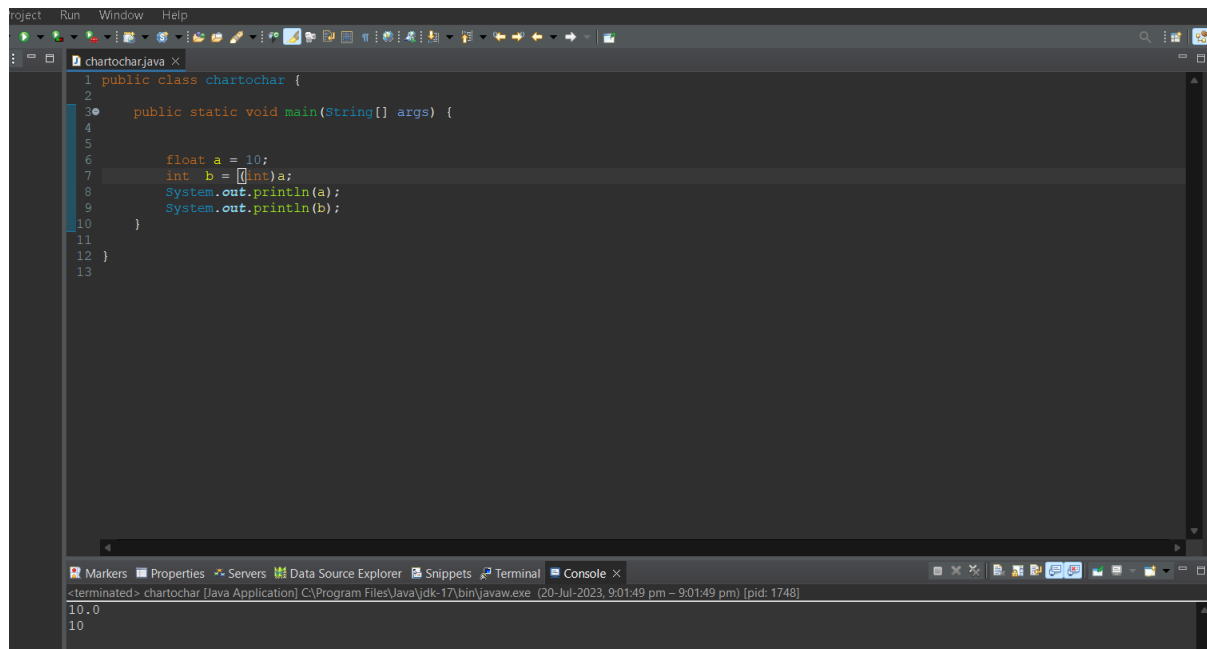
Markers Properties Servers Data Source Explorer Snippets Terminal Console

<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 9:00:35 pm - 9:00:35 pm) [pid: 19788]

10.0
10

4) Float to int explicit :

In Java, you can convert a **float** data type to an **int** data type using type casting. However, be aware that converting a **float** to an **int** may result in a loss of precision, as **int** is a 32-bit signed integer data type, and **float** is a 32-bit floating-point data type.

A screenshot of an IDE window titled 'chartochar.java'. The code defines a class 'chartochar' with a 'main' method. Inside 'main', a float variable 'a' is assigned the value 10.0. This float is then explicitly cast to an integer and stored in variable 'b' using the syntax '(int)a'. Finally, 'a' and 'b' are printed to the console. The IDE's console at the bottom shows the output: '10.0' followed by '10'.

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10.0;
6         int b = (int)a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console

<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 9:01:49 pm - 9:01:49 pm) [pid: 1748]

10.0
10

5) Float to long explicit:

In Java, you can convert a **float** data type to a **long** data type using type casting. However, be aware that converting a **float** to a **long** may result in a loss of precision, as **long** is a 64-bit integer data type, and **float** is

Example:

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10.0;
6         long b = (long) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

Console output: 10.0, 10

32-bit floating-point data type.

6) Float to double implicit:

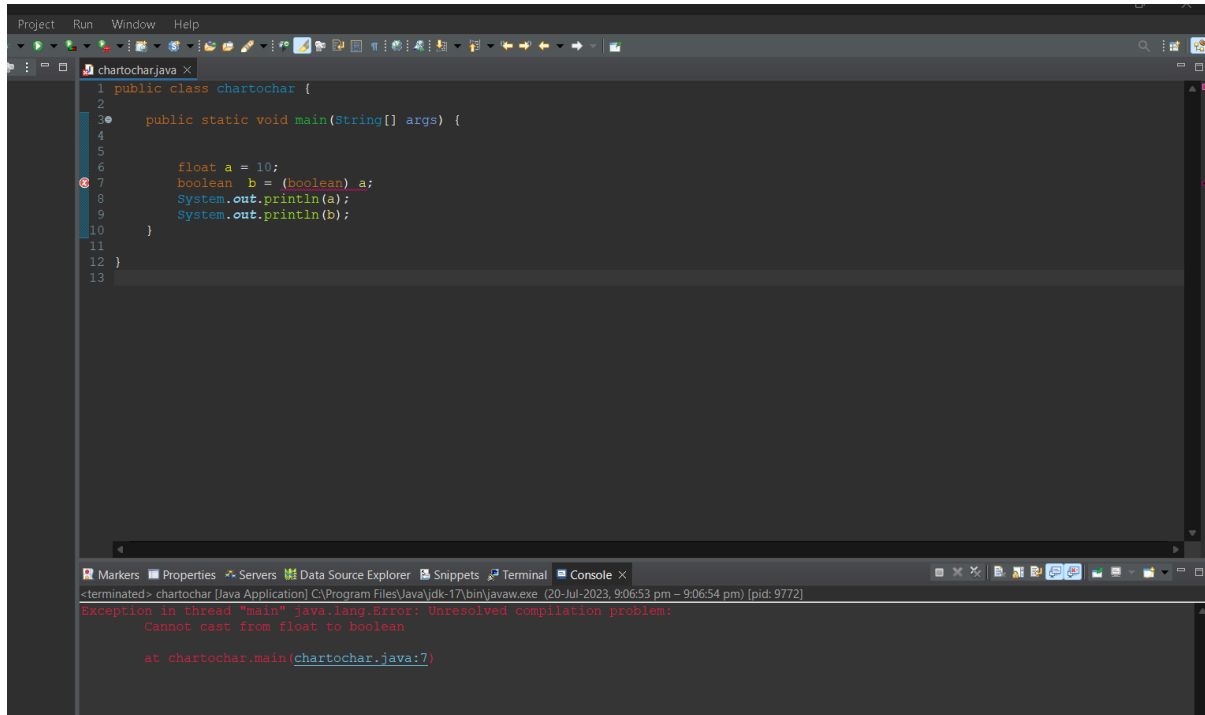
In Java, you can convert a **float** data type to a **double** data type without any explicit type casting. This is because **double** is a higher precision floating-point data type than **float**, and Java allows implicit widening conversions from **float** to **double**.

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10.0;
6         double b = a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

Console output: 10.0, 10.0

7) Float to Boolean explicit:

In Java, there is no direct conversion from a **float** data type to a **boolean** data type. **boolean** can only represent two values: **true** or **false**. A **float** is a numeric data type and cannot be directly converted to a boolean value.



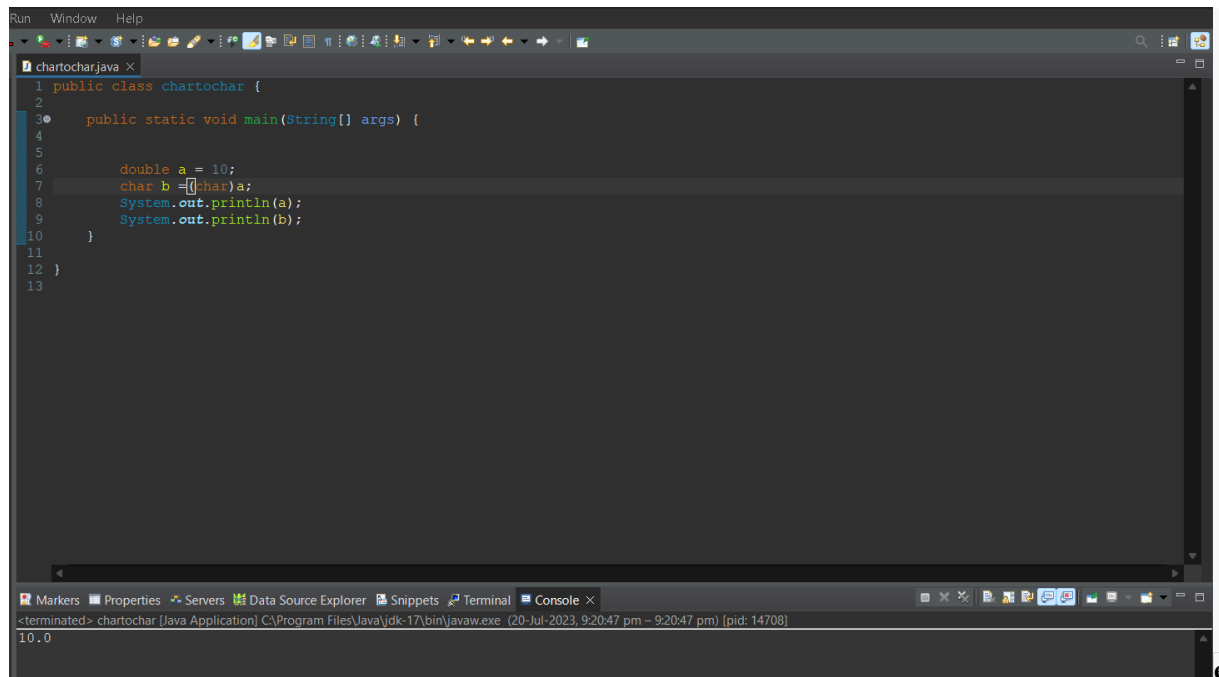
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         float a = 10;
6         boolean b = (boolean) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10
11 }
12
13
```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Cannot cast from float to boolean
at chartochar.main(chartochar.java:7)

1) Double to char explicit:

In Java, you can convert a **double** data type to a **char** data type by first converting the **double** to an integer and then converting the integer to a **char**. This approach assumes that the **double** value represents a valid Unicode value for the **char**.

Example:

A screenshot of an IDE window titled 'charToChar.java'. The code is as follows:

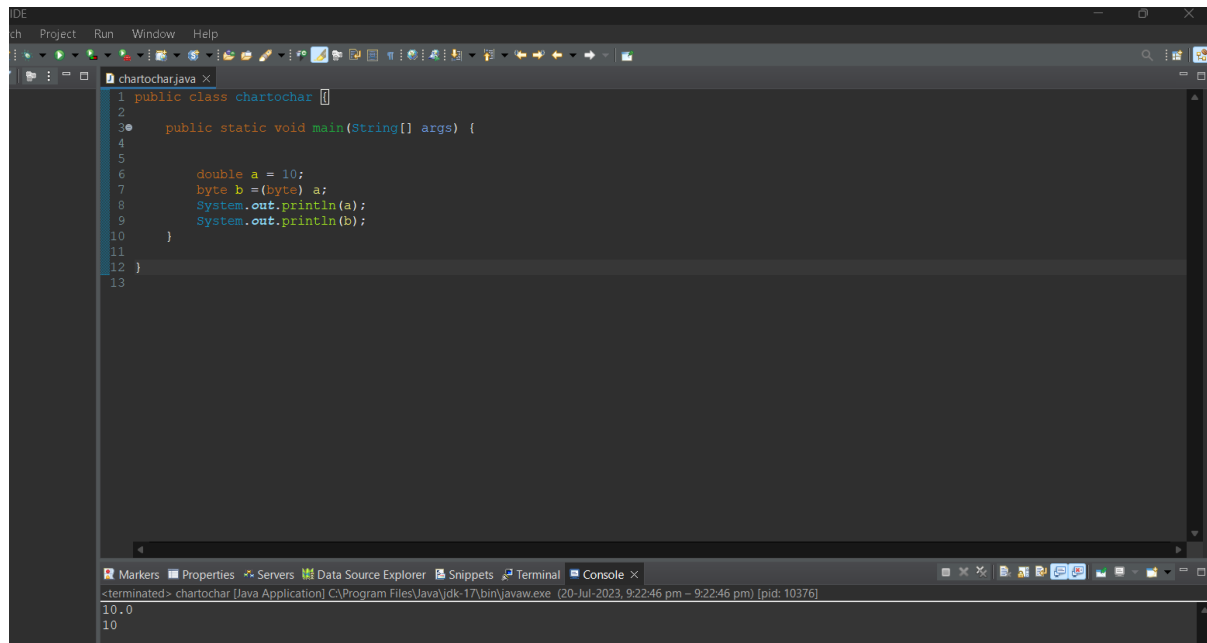
```
1 public class charToChar {  
2  
3     public static void main(String[] args) {  
4  
5         double a = 10;  
6         char b = (char)a;  
7         System.out.println(a);  
8         System.out.println(b);  
9     }  
10 }  
11  
12 }  
13
```

The IDE's console at the bottom shows the output of the program: '10.0' on the first line and 'e' on the second line. The console title is 'Console x' and the text below it reads: '<terminated> charToChar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 9:20:47 pm - 9:20:47 pm) [pid: 14708] 10.0'.

2) Double to byte explicit:

In Java, you can convert a **double** data type to a **byte** data type using type casting. However, be aware that converting a **double** to a **byte** may result in a loss of precision, as **byte** is an 8-bit signed integer data type, and **double** is a 64-bit floating-point data type.

Example:



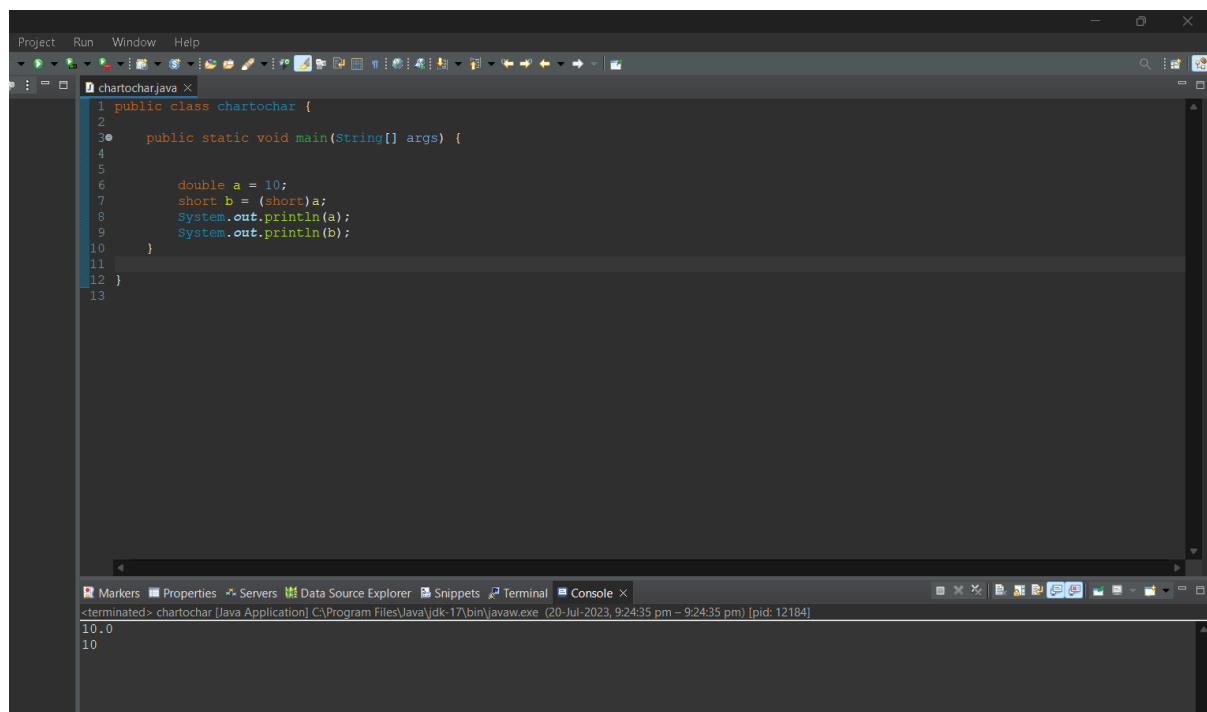
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         double a = 10;
6         byte b = (byte) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The console output shows the value 10.0 for the double variable and 10 for the byte variable, indicating a successful cast without loss of precision in this specific case.

3) Double to short explicit :

In Java, you can convert a **double** data type to a **short** data type using type casting. However, be aware that converting a **double** to a **short** may result in a loss of precision, as **short** is a 16-bit signed integer data type, and **double** is a 64-bit floating-point data type.

Example:



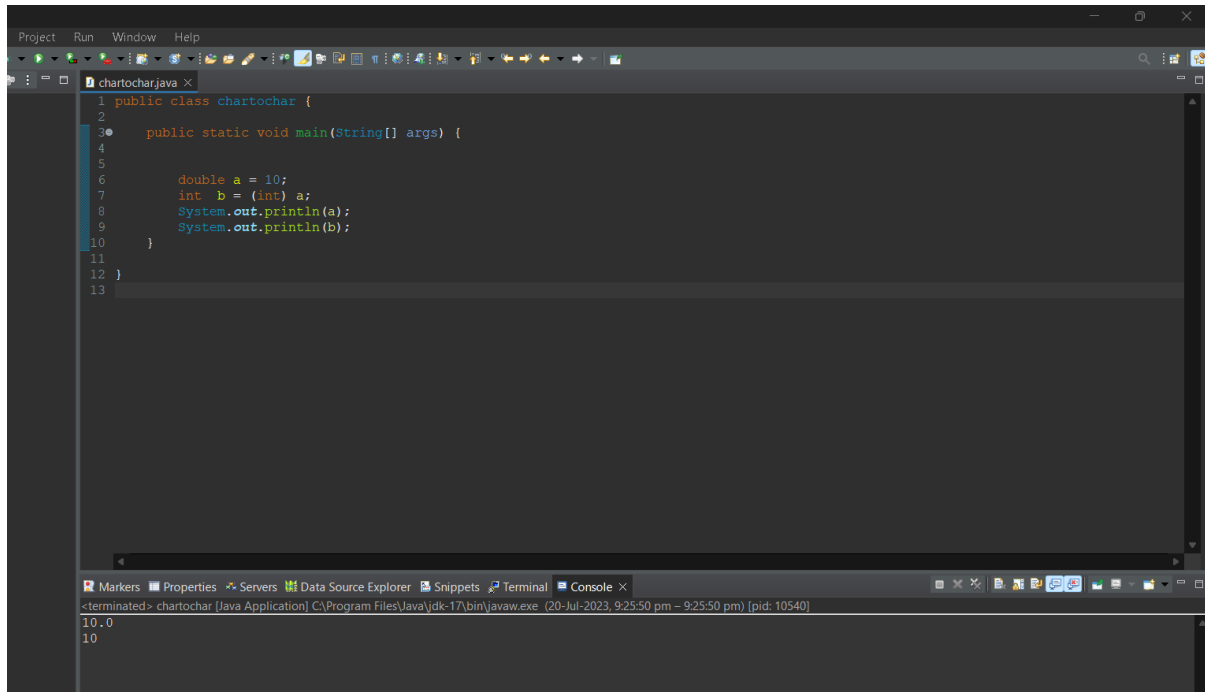
```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         double a = 10;
6         short b = (short)a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

The console output shows the value 10.0 for the double variable and 10 for the short variable, indicating a successful cast without loss of precision in this specific case.

4) Double to int explicit:

In Java, you can convert a **double** data type to an **int** data type using type casting. However, be aware that converting a **double** to an **int** may result in a loss of precision, as **int** is a 32-bit signed integer data type, and **double** is a 64-bit floating-point data type.

Example:

A screenshot of an IDE window titled 'chartochar.java'. The code defines a class 'chartochar' with a 'main' method. Inside 'main', a 'double' variable 'a' is assigned the value 10.0, and an 'int' variable 'b' is assigned the value of 'a' after explicit casting: 'int b = (int) a;'. Both variables are printed to the console. The IDE's console at the bottom shows the output: '10.0' followed by '10'.

```
1 public class chartochar {  
2  
3     public static void main(String[] args) {  
4  
5         double a = 10.0;  
6         int b = (int) a;  
7         System.out.println(a);  
8         System.out.println(b);  
9     }  
10 }  
11  
12  
13
```

<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 9:25:50 pm - 9:25:50 pm) [pid: 10540]
10.0
10

5) Double to long explicit:

In Java, you can convert a **double** data type to a **long** data type using type casting. However, be aware that converting a **double** to a **long** may result in a loss of precision, as **long** is a 64-bit integer data type, and **double** is a 64-bit floating-point data type.

Example:

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         double a = 10;
6         long b = (long) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

Console output:

```
<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 9:27:28 pm - 9:27:28 pm) [pid: 20788]
10.0
10
```

6) Double to float explicit :

In Java, you can convert a **double** data type to a **float** data type using type casting. This conversion is called narrowing conversion, and you may lose precision when converting from **double** to **float**, as **double** is a 64-bit floating-point data type and **float** is a 32-bit floating-point data type.

Example:

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5         double a = 10;
6         float b = (float) a;
7         System.out.println(a);
8         System.out.println(b);
9     }
10 }
11
12
13
```

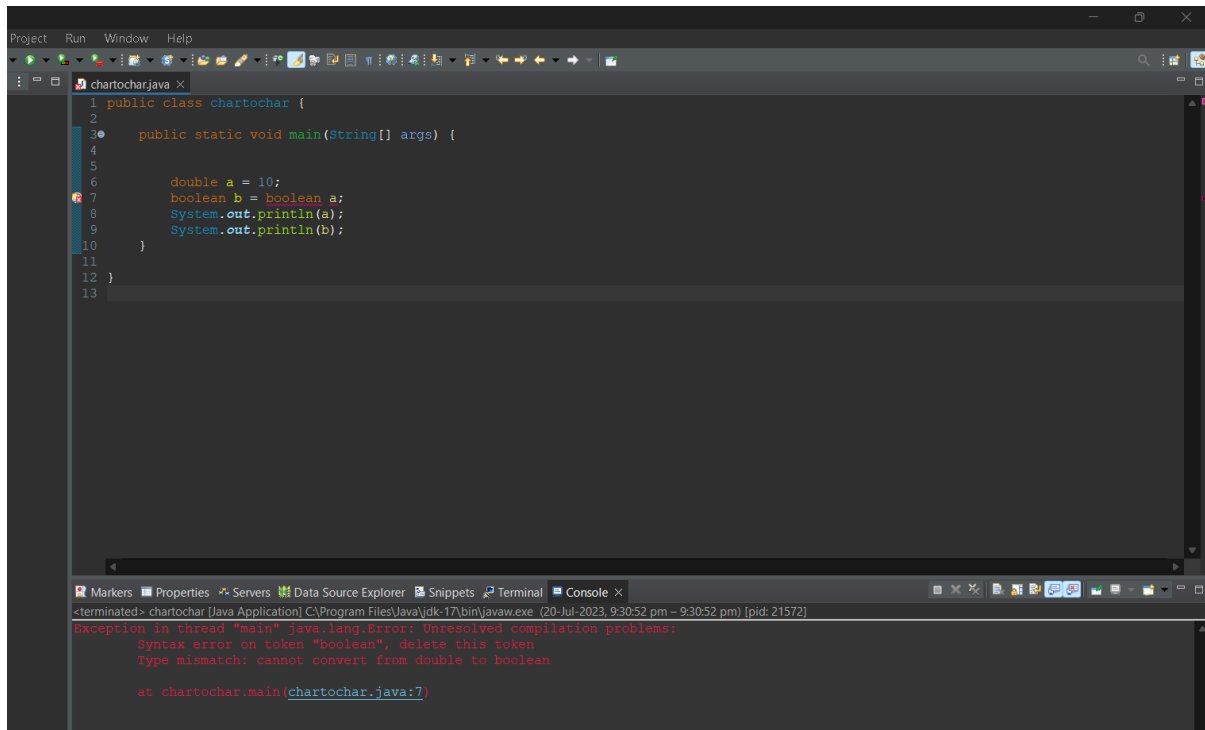
Console output:

```
<terminated> chartochar [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (20-Jul-2023, 9:28:58 pm - 9:28:58 pm) [pid: 13560]
10.0
10.0
```

7) Double to Boolean explicit:

In Java, you cannot directly convert a **double** data type to a **boolean** data type using type casting, as there is no direct numeric representation of **true** or **false** in a **double**.

Example:

A screenshot of an IDE window showing a Java file named 'chartochar.java'. The code defines a public class 'chartochar' with a main method. Inside the main method, a double variable 'a' is assigned the value 10. Then, a boolean variable 'b' is assigned the value of 'boolean a;', which is a syntax error. The IDE's console shows a 'java.lang.Error: Unresolved compilation problems: Syntax error on token "boolean", delete this token. Type mismatch: cannot convert from double to boolean' at line 7 of 'chartochar.java'.

```
1 public class chartochar {
2
3     public static void main(String[] args) {
4
5
6         double a = 10;
7         boolean b = boolean a;
8         System.out.println(a);
9         System.out.println(b);
10    }
11 }
12 }
13 }
```

Exception in thread "main" java.lang.Error: Unresolved compilation problems:
Syntax error on token "boolean", delete this token
Type mismatch: cannot convert from double to boolean
at chartochar.main(chartochar.java:7)

- 1) Boolean to char explicit, 2) 1) Boolean to short explicit, 3) Boolean to int explicit,
4) Boolean to long explicit, 5) Boolean to float explicit, 6) Boolean to double explicit,
7) Boolean to byte explicit,

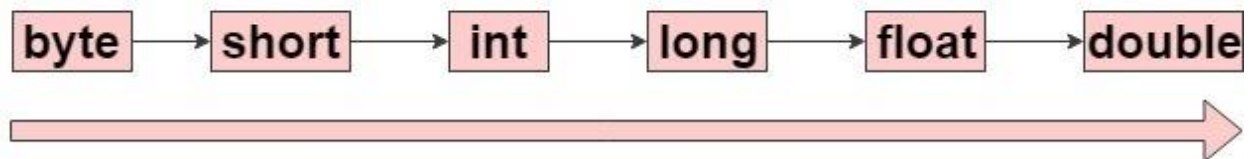
In Java, you cannot directly convert a **boolean** data type to a **any** data type using type casting, as there is no direct numeric representation of **true** or **false** in a any datatype.

Table for all datatype conversions

	byte	short	char	int	long	float	double
byte		expl	expl	expl	expl	expl	expl
short	impl		expl	expl	expl	expl	expl
char	expl	expl		expl	expl	expl	expl
int	impl	impl	impl		expl	expl	expl
long	impl	impl	impl	impl		expl	expl
float	impl	impl	impl	impl	impl		expl
double	impl	impl	impl	impl	impl	impl	

Datatype conversion graph

Automatic Type Conversion (Widening - implicit)



Narrowing (explicit)

