

Forum:[Hack Forums](#)

Thread: [\[char cannot be converted to String\] bypass?](#)

SoyNuevo - November 13, 2016, 6:46 pm

Hey, so you can't do stuff like

`Code Integer.parseInt(string.charAt(i));` because the parse method requires a string instead of a char
.

However, if you do this:

`Code Integer.parseInt("" + string.charAt(i));` java thinks it is perfectly acceptable and accepts the char. because java reads left -> right

I'm wondering if this is good practice or if there is a specific way I should be converting my chars to strings. thanks

FinnTheHuman - November 13, 2016, 6:59 pm

In order to convert a char to a string you can simply use

```
String myString = new String(myChar);
```

Stealth™ - November 13, 2016, 7:07 pm

OP, the method you currently use to parse a char into a String is an anti-pattern and should be avoided.

A proper way would consist of calling `Character.toString`.

FeatureBug - November 13, 2016, 11:42 pm

(Nov 13, 2016 07:46 PM)SoyNuevo Wrote: [\[CITE\]](#) `Code Integer.parseInt("" + string.charAt(i));`
Typically, you never want to be concatenating strings this way, even if it's just for a little hack-together like this.

(Nov 13, 2016 08:07 PM)Stealth™ Wrote: [\[CITE\]](#) A proper way would consist of calling `Character.toString`.

This is the best way to achieve what you want.

Do you use an IDE to develop in? I'm surprised it didn't recommend that function.

Intel - November 14, 2016, 4:10 pm

(Nov 13, 2016 07:46 PM)SoyNuevo Wrote: [\[CITE\]](#) Hey, so you can't do stuff like
`Code Integer.parseInt(string.charAt(i));` because the parse method requires a string instead of a char
.

However, if you do this:

`Code Integer.parseInt("" + string.charAt(i));` java thinks it is perfectly acceptable and accepts the char. because java reads left -> right

I'm wondering if this is good practice or if there is a specific way I should be converting my chars to strings. thanks

Just from what you're trying to do, converting an ascii char into an int:

PHP Code

```
<?php
```

```
int x = (int) (string.charAt(i) - '0');
```

This would be a faster solution to what you're trying to do.

Stealth™ - November 14, 2016, 7:42 pm

(Nov 14, 2016 05:10 PM)Synodic Wrote: [\[CITE\]](#) Just from what you're trying to do, converting an ascii char into an int:

PHP Code

```
<?php
```

```
int x = (int) (string.charAt(i) - '0');
```

This would be a faster solution to what you're trying to do.

That's even worse than what OP has

TCP - November 15, 2016, 2:46 am

You can even just do

```
Code Integer.parseInt(string.substring(i,i+1));
```

Not2EXceL - November 15, 2016, 2:34 pm

(Nov 15, 2016 03:46 AM)KitKat707 Wrote: [\[CITE\]](#) You can even just do

```
Code Integer.parseInt(string.substring(i,i+1));
```

Accessing the underlying char array via `String#charAt(int)` and converting via `String#valueOf(char)` or `Character#toString(char)` is faster

TCP - November 15, 2016, 3:06 pm

(Nov 15, 2016 03:34 PM)Not2EXceL Wrote: [\[CITE\]](#) Accessing the underlying char array via `String#charAt(int)` and converting via `String#valueOf(char)` or `Character#toString(char)` is faster
I am getting different results with this

```
Code String string = "number 5 is cool";
```

```
int i = 7, x;
```

```
long time = System.nanoTime();
```

```
x = Integer.parseInt(string.substring(i,i+1));
```

```
System.out.println(System.nanoTime()-time);
```

```
time = System.nanoTime();
```

```
x = Integer.parseInt(Character.toString(string.charAt(i)));
```

```
System.out.println(System.nanoTime()-time);
```

The first version is running faster with Java 8+ on my pc but when I put it online it is slower

matt - November 15, 2016, 8:11 pm

(Nov 15, 2016 04:06 PM)KitKat707 Wrote: [\[CITE\]](#) I am getting different results with this

```
Code String string = "number 5 is cool";
```

```
int i = 7, x;
```

```
long time = System.nanoTime();
x = Integer.parseInt(string.substring(i,i+1));
System.out.println(System.nanoTime()-time);
```

```
time = System.nanoTime();
x = Integer.parseInt(Character.toString(string.charAt(i)));
System.out.println(System.nanoTime()-time);
```

The first version is running faster with Java 8+ on my pc but when I put it online it is slower there's no need to micro-optimize, just keep it simple and up to date procedure.

Chickendipper - November 16, 2016, 4:10 am

(Nov 15, 2016 04:06 PM)KitKat707 Wrote: [CITE](#) I am getting different results with this

```
Code String string = "number 5 is cool";
int i = 7, x;
```

```
long time = System.nanoTime();
x = Integer.parseInt(string.substring(i,i+1));
System.out.println(System.nanoTime()-time);
```

```
time = System.nanoTime();
x = Integer.parseInt(Character.toString(string.charAt(i)));
System.out.println(System.nanoTime()-time);
```

The first version is running faster with Java 8+ on my pc but when I put it online it is slowerThis is literally the worst benchmark in all of existence.

You're doing 1 iteration

You only ran it once(?)

The JVM will not have had time to make necessary optimisations

You're comparing run times from your computer to a fully loaded online java compiler

For more, [read this](#) and then create a proper benchmark.

@OP: "" + char is bad practice. Why? When you do:

```
string1 + string2
```

in Java, what's actually happening (when compiled, under the hood) is:

```
new StringBuilder().append(string1).append(string2).toString()
```

So we're creating a new StringBuilder object. This takes up memory, so we're using up more memory than we wanted (yes, the GC will eventually get to it, but it's still bad practice).

Furthermore, doing:

```
string1 + object1 (where object1 is anything BUT a string)
```

in Java, this happens instead:

```
new StringBuilder().append(string1).append(object1.toString()).toString()
```

Thus we're creating a new object and invoking the #toString() method created by the superclass of object1. We're using up a lot more resources than we need.

The preferred solution would be to, for example, use Character.toString(char).

TCP - November 16, 2016, 4:40 am

(Nov 16, 2016 05:10 AM)Chickendipper Wrote: [\[CITE\]](#) This is literally the worst benchmark in all of existence.

Was not a benchmark just a test, every time I ran it myself the first solution was faster, same with multiple iterations in a loop

My point is there is no really a faster solution as the other guy said because it depends on environment

.charAt() is better for readability in such a context