How to cast an Object to an int

196

43

How can I cast an Object to an int in java?

java object casting integer

edited Jan 27 '18 at 21:48

Taslim Oseni
1,536 4 17 29

asked Sep 7 '10 at 18:15

Sheehan Alam
26.8k 117 334 541

2 What kind of Object? – NullUserException Sep 7 '10 at 18:17

6 What do you really want to do? If the `Object` isn't an `Integer`, I'm not sure what your are expecting from your cast. – unholysampler Sep 7 '10 at 18:19

2 first check with instanceof keyword . if true then cast it. – Dead Programmer Sep 8 '10 at 11:04

Aww. I just wanted to have enum members to cast to specific integer values, so that I can have enums for winapi constants. msdn.microsoft.com/en-us/library/windows/desktop/… – Tomáš Zato Jan 27 '15 at 19:29

@TomášZato You can do that (sort of), just define a field in your enum to hold the integer value (say, `intValue`), create a constructor for your enum that sets the `intValue`, have your enum constants invoke that constructor, and add a getter for `intValue`. Then, instead of casting, call the getter. – Brian McCutchon May 23 '16 at 18:12

18 Answers

351

If you're sure that this object is an `Integer`:

```
int i = (Integer) object;
```

Or, starting from Java 7, you can equivalently write:

Beware, it can throw a `ClassCastException` if your object isn't an `Integer` and a `NullPointerException` if your object is `null`.

This way you assume that your Object is an Integer (the wrapped int) and you unbox it into an int.

`int` is a primitive so it can't be stored as an `Object`, the only way is to have an `int` considered/boxed as an `Integer` then stored as an `Object`.

If your object is a `String`, then you can use the `Integer.valueOf()` method to convert it into a simple int :

```
int i = Integer.valueOf((String) object);
```

It can throw a `NumberFormatException` if your object isn't really a `String` with an integer as content.

Resources :

- Oracle.com - Autoboxing
- Oracle.com - Primitive Data types

On the same topic :

- Java: What's the difference between autoboxing and casting?
- Autoboxing: So I can write: Integer i = 0; instead of: Integer i = new Integer(0);
- Convert Object into primitive int

edited May 23 '17 at 11:54
Community ♦
1 1

answered Sep 7 '10 at 18:19
Colin Hebert
77.1k 12 139 137

Are you sure about the NullPointerException? I thought that a null Object would just yield a null Integer.... – Etienne de Martel Sep 7 '10 at 18:23

7 The `NullPointerException` will occur during the unboxing of `Integer` into `int` – Colin Hebert Sep 7 '10 at 18:24

1 Ah, yeah, my brain ignored the left part of the assignment... – Etienne de Martel Sep 7 '10 at 18:27

You're not casting to an int, no Object can ever be cast to an int. You're actually to Integer and then autoboxing to an int. – Steve Kuo Sep 7 '10 at

17

Assuming the object is an `Integer` object, then you can do this:

```
int i = ((Integer) obj).intValue();
```

If the object isn't an `Integer` object, then you have to detect the type and convert it based on its type.

answered Sep 7 '10 at 18:17

Erick Robertson
25.5k 7 63 93

If obj is null it will throw a NullPointerException. – Colin Hebert Sep 7 '10 at 18:25

and a ClassCastException if it's not an Integer object. – Erick Robertson Sep 7 '10 at 18:27

1 No need to invoke `intValue` for autoboxing will invoke it for you. – OscarRyz Sep 7 '10 at 18:35

1 `intValue` is much clearer especially considering the beginner confusion between `int` being interchangeable with `Integer` . – Steve Kuo Sep 7 '10 at 19:18

11

```
@Deprecated
public static int toInt(Object obj)
{
    if (obj instanceof String)
    {
        return Integer.parseInt((String) obj);
    } else if (obj instanceof Number)
    {
        return ((Number) obj).intValue();
    } else
    {
        String toString = obj.toString();
        if (toString.matches("-?\d+"))
        {
            return Integer.parseInt(toString);
        }
```

As you can see, this isn't a very efficient way of doing it. You simply have to be sure of what kind of object you have. Then convert it to an int the right way.

edited Dec 1 '16 at 15:49
tuckerpm
108 1 9

answered Sep 7 '10 at 18:51

Martijn Courteaux
50.3k 38 174 263

Isn't it @Deprecated (e in stead of a)? :) Nice method though, makes no assumptions on the type of the object. – extraneon Sep 8 '10 at 7:25

By the way, your regex doesn't take radix hex or oct into account. ToInt is a smart method. Bettere to try and catch NumberFormatExcepytion. – extraneon Sep 8 '10 at 16:43

11

Scenario 1: simple case

If it's guaranteed that your object is an `Integer`, this is the simple way:

```
int x = (Integer)yourObject;
```

Scenario 2: any numerical object

In Java `Integer`, `Long`, `BigInteger` etc. all implement the `Number` interface which has a method named `intValue`. Any other custom types with a numerical aspect should also implement `Number` (for example: `Age implements Number`). So you can:

```
int x = ((Number)yourObject).intValue();
```

Scenario 3: parse numerical text

When you accept user input from command line (or text field etc.) you get it as a `String`. In this case you can use `Integer.parseInt(String string)`:

```
String input = someBuffer.readLine();
```

If you get input as `Object`, you can use `(String)input`, or, if it can have an other textual type, `input.toString()`:

```
int x = Integer.parseInt(input.toString());
```

Scenario 4: identity hash

In Java there are no pointers. However `Object` has a pointer-like default implementation for `hashCode()`, which is directly available via `System.identityHashCode(Object o)`. So you can:

```
int x = System.identityHashCode(yourObject);
```

Note that this is **not** a real pointer value. Objects' memory address can be changed by the JVM while their identity hashes are keeping. Also, two living objects can have the same identity hash.

You can also use `object.hashCode()`, but it can be type specific.

Scenario 5: unique index

In same cases you need a unique index for each object, like to auto incremented ID values in a database table (and unlike to identity hash which is not unique). A simple sample implementation for this:

```
class ObjectIndexer {

    private int index = 0;

    private Map<Object, Integer> map = new WeakHashMap<>();

    public int indexFor(Object object) {
        if (map.containsKey(object)) {
            return map.get(object);
        } else {
            index++;
            map.put(object, index);
            return index;
        }
    }

}
```

```
ObjectIndexer indexer = new ObjectIndexer();
int x = indexer.indexFor(yourObject);    // 1
int y = indexer.indexFor(new Object());  // 2
int z = indexer.indexFor(yourObject);    // 1
```

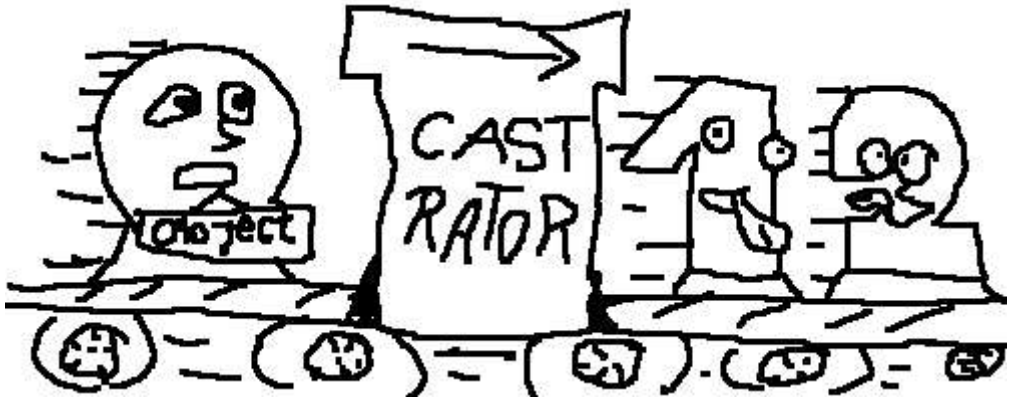
Scenario 6: enum member

In Java enum members aren't integers but full featured objects (unlike C/C++, for example). Probably there is never a need to convert an enum object to `int`, however Java automatically associates an index number to each enum member. This index can be accessed via `Enum.ordinal()`, for example:

```
enum Foo { BAR, BAZ, QUX }

// ...

Object baz = Foo.BAZ;
int index = ((Enum)baz).ordinal(); // 1
```



edited Jun 12 '18 at 14:57

answered Oct 16 '15 at 13:54

Dávid Horváth

2,479 1 15 27

4

answered Sep 7 '10 at 18:17

Etienne de Martel
26.2k 6 76 94

4

Answer:

```
int i = ( Integer ) yourObject;
```

If, your object is an integer already, it will run smoothly. ie:

```
Object yourObject = 1;
//  cast here
```

or

```
Object yourObject = new Integer(1);
//  cast here
```

etc.

If your object is anything else, you would need to convert it (if possible) to an int first:

```
String s = "1";
Object yourObject = Integer.parseInt(s);
//  cast here
```

Or

```
String s = "1";
Object yourObject = Integer.valueOf( s );
//  cast here
```

answered Sep 7 '10 at 18:33

I use a one-liner when processing data from GSON:

3

```
int i = object != null ? Double.valueOf(object.toString()).intValue() : 0;
```

edited Dec 12 '13 at 22:22

answered Dec 12 '13 at 22:00

Someone Somewhere
19.2k 10 97 144

Its a lengthy process. Just do **(int)Object** instead of **Double.valueOf(object.toString()).intValue()**. This works only for numbers, thats what we needed. – Sudhakar Krishnan Feb 16 '14 at 13:03

1 @SudhakarK: (int) Object does only work if your object is a Integer. This oneliner also supports String numbers; E.G. "332". – Jacob van Lingen Aug 8 '14 at 7:18

2

If the `Object` was originally been instantiated as an `Integer`, then you can downcast it to an `int` using the cast operator `(Subtype)`.

```
Object object = new Integer(10);
int i = (Integer) object;
```

Note that this only works when you're using at least Java 1.5 with autoboxing feature, otherwise you have to declare `i` as `Integer` instead and then call `intValue()` on it.

But if it initially wasn't created as an `Integer` at all, then you can't downcast like that. It would result in a `ClassCastException` with the original classname in the message. If the object's `toString()` representation as obtained by `String#valueOf()` denotes a syntactically valid integer number (e.g. digits only, if necessary with a minus sign in front), then you can use `Integer#valueOf()` or `new Integer()` for this.

```
Object object = "10";
int i = Integer.valueOf(String.valueOf(object));
```

See also:

edited Sep 7 '10 at 18:32

answered Sep 7 '10 at 18:20

BalusC
873k 309 3225 3267

2

```
int i = (Integer) object; //Type is Integer.

int i = Integer.parseInt((String)object); //Type is String.
```

answered Sep 7 '10 at 18:36

Kerem Baydoğan
8,850 32 47

2

Can't be done. An `int` is not an object, it's a primitive type. You can cast it to Integer, then get the int.

```
Integer i = (Integer) o; // throws ClassCastException if o.getClass() != Integer.class

int num = i; //Java 1.5 or higher
```

edited Sep 8 '10 at 13:11

answered Sep 7 '10 at 18:18

Tom
31.2k 23 120 157

This assumes that the object is an integer which it almost certainly is not. Probably want's the string solution ala Coronauts – Bill K Sep 7 '10 at 18:20

And won't compile. – Ricky Clarkson Sep 8 '10 at 6:28

@Ricky What part? 1.4, 1.5? – Tom Sep 8 '10 at 12:22

How could it compile when you are casting an object into Object and then trying to set it to an Integer variable. – Carlos Sep 8 '10 at 12:48

2

edited Jun 14 '16 at 11:58

Alexander Farber
8,228 56 191 336

answered Sep 7 '10 at 18:17

extraneon
19.9k 2 38 49

There's auto boxing/unboxing since Java 5. – Bruno Sep 7 '10 at 18:19

@Bruno: You can't cast an Object to an int. You can cast an Object to an Integer and then assign it to an int and it will magically autounbox. But you can't cast an Object to an int. – Jay Sep 7 '10 at 21:05

(continued) Personally, I think people create a lot of bad code relying on autoboxing. Like, I saw a statement the other day, "Double amount= (Double.parseDouble(stringAmount)).doubleValue();". That is, he parsed a String to get a double primitive, then executed a function against this, which forced the compiler to autobox it into a Double object, but the function was doubleValue which extracted the double primitive, which he then assigned to a Double object thus forcing an autobox. That is, he converted from primitive to object to primitive to object, 3 conversions. – Jay Sep 7 '10 at 21:07

@Jay, agreed on 1st comment (sorry I wasn't clear myself). Regarding too many conversion, you're right too, but I get the impression that the JIT compiler can cope with that quite well, so it shouldn't matter that much in practice (that doesn't necessarily make it an excuse for bad code...) – Bruno Sep 7 '10 at 22:30

1 @Bruno The trickypart of autoboxing it that it can give you unexpected NullPointerExceptions. – extraneon Sep 8 '10 at 7:24

1

If you mean cast a String to int, use `Integer.valueOf("123")` .

You can't cast most other Objects to int though, because they wont have an int value. E.g. an XmlDocument has no int value.

answered Sep 7 '10 at 18:18

Amy B
14.2k 12 58 77

1 Don't use `Integer.valueOf("123")` if all you need is a primitive instead use `Integer.parseInt("123")` because **valueOf** method causes an unnecessary unboxing. – Kerem Baydoğan Sep 7 '10 at 18:32

1

I guess you're wondering why C or C++ lets you manipulate an object pointer like a number, but you can't manipulate an object reference in Java the same way.

Object references in Java aren't like pointers in C or C++. Pointers basically are integers and you can manipulate them like any other

answered Sep 7 '10 at 19:15

romacafe

1,995 2 19 25

1

```
int[] getAdminIDList(String tableName, String attributeName, int value) throws
SQLException {
    ArrayList list = null;
    Statement statement = conn.createStatement();
    ResultSet result = statement.executeQuery("SELECT admin_id FROM " + tableName + "
WHERE " + attributeName + "='" + value + "'");
    while (result.next()) {
        list.add(result.getInt(1));
    }
    statement.close();
    int id[] = new int[list.size()];
    for (int i = 0; i < id.length; i++) {
        try {
            id[i] = ((Integer) list.get(i)).intValue();
        } catch(NullPointerException ne) {
        } catch(ClassCastException ch) {}
    }
    return id;
}
// enter code here
```

This code shows why `ArrayList` is important and why we use it. Simply casting `int` from `Object` . May be its helpful.

edited May 21 '14 at 6:58

Harmlezz

6,904 19 30

answered May 21 '14 at 6:30

Mahbubur Rahman Khan

132 15

Please explain your answer – Gwenc37 May 21 '14 at 6:52

For Example Object variable; `hastaId`

For Example Cast an Object to an int, `hastaID`

```
int hastaID=Integer.parseInt(String.valueOf(hastaId));
```

edited Feb 28 '18 at 9:58

WhatsThePoint
2,381 6 22 39

answered Dec 24 '14 at 14:59

user4391934
11 1

0

Refer This code:

```
public class sample
{
  public static void main(String[] args)
  {
    Object obj=new Object();
    int a=10,b=0;
    obj=a;
    b=(int)obj;

    System.out.println("Object="+obj+"\nB="+b);
  }
}
```

answered Oct 2 '14 at 12:57

Vishal Tathe
152 1 8

0

map.getValue() returns object type so divide1=me.getValue()/2;

int divide1 = (Integer) me.getValue()/2;

answered Jan 27 at 20:45

Nehal Pawar

What does this add that isn't already covered in the existing answers? – Robert Jan 27 at 21:42

This shows a situation where the casting is required and I will add the error as well that actually shows up with this situation. Its hard for a new coder to figure out the actual implementation if there is no example. I hope this example helps them – Nehal Pawar Jan 28 at 22:39

-3

first check with instanceof keyword . if true then cast it.

answered Sep 8 '10 at 11:06

Dead Programmer
9,052 19 69 107