

Cast, Chop, and Wrap

Casting and range of variables



Numbers with limits

PAUSE AND TRY

What happens in each case? Talk it over with a partner.

ODOMETER

9999999

You drive one more mile. What does it show now?

YOUR GRADE

89.6

The report card shows whole numbers only. 89 or 90?

By the end of class you can...

- 1 Explain when Java widens an int to a double on its own.
- 2 Cast with (int) and (double), and predict the value: (int) truncates.
- 3 Put a cast where it applies to the value you mean.
- 4 Round a double to the nearest int, for positive and negative numbers.
- 5 Explain the range of int and what integer overflow does.
- 6 Explain round-off error, and when to use int to avoid it.

1

PART 1

From int to double and back

Java widens an int to a double for you



Some code causes int values to be automatically cast, or **widened**, to double values. The amount stays the same; only the type changes.

WIDENING AT WORK

```
double price = 5;  
System.out.println(price);  
int laps = 3;  
double miles = laps * 0.25;  
System.out.println(miles);
```

OUTPUT

```
5.0  
0.75
```

Where it happens	Example
Storing an int in a double variable	<code>double price = 5;</code>
An operation with one int and one double	<code>laps * 0.25</code>

Casting a double to an int chops the decimal



The casting operators **(int)** and **(double)** convert a value. Casting a double to an int **truncates**: the digits right of the decimal point are dropped.

TRUNCATING

```
System.out.println((int) 26.2);  
System.out.println((int) 26.9);  
System.out.println((int) -3.8);
```

OUTPUT

```
26  
26  
-3
```

WITHOUT A CAST

```
int miles = 26.2;
```

COMPILER SAYS

```
incompatible types: possible lossy  
conversion from double to int
```

Java will not narrow on its own.

A cast applies only to the value right after it



Cast **before** you divide. Casting the answer of an int division is too late: the decimal part is already gone.

```
int total = 22;  
int count = 5;  
System.out.println((double) total / count);  
System.out.println((double) (total / count));
```

Line 3: 22.0 / 5, double division, gives 4.4

Line 4: 22 / 5 is int division, 4; then the cast makes 4.0

OUTPUT

4.4

4.0

What prints?

PAUSE AND TRY

Predict all three lines of output.

```
double hours = 7.75;  
int whole = (int) hours;  
double back = whole;  
System.out.println(whole);  
System.out.println(back);  
System.out.println(hours);
```

THINK ABOUT

Does line 3 get the .75 back?

Does the cast on line 2 change hours?

A large, white, rounded square containing the number '2' in a bold, teal, sans-serif font.

2

PART 2

Rounding to the nearest int

Add 0.5, then chop



For a value x that is zero or positive, `(int) (x + 0.5)` is x rounded to the nearest integer.

x	$x + 0.5$	Truncated
8.6	9.1	9
8.4	8.9	8
8.5	9.0	9

```
double x = 8.6;  
System.out.println((int) (x + 0.5));  
x = 8.4;  
System.out.println((int) (x + 0.5));  
x = 8.5;  
System.out.println((int) (x + 0.5));
```

OUTPUT

```
9  
8  
9
```

Rounding a negative number



For a negative value x , use `(int) (x - 0.5)` to round to the nearest integer.

A TEMPERATURE DROP

```
double drop = -6.7;  
System.out.println((int) (drop + 0.5));  
System.out.println((int) (drop - 0.5));
```

OUTPUT

-6

-7

Expression	Before cast	Result
<code>drop + 0.5</code>	-6.2	-6 (wrong)
<code>drop - 0.5</code>	-7.2	-7 (right)

-6.7 is closer to -7 on a number line.

Keep the parentheses around the sum



A cast applies to the value right after it. Without parentheses, Java casts `x` and **then** adds `0.5`.

DOES NOT COMPILE

```
double x = 8.6;  
int r = (int) x + 0.5;
```

COMPILER SAYS

```
incompatible types: possible lossy  
conversion from double to int
```

FIXED

```
double x = 8.6;  
int r = (int) (x + 0.5);  
System.out.println(r);
```

OUTPUT

9

Round these

PAUSE AND TRY

Pick + 0.5 or - 0.5 for each value, then write the rounded int.

x	Expression to use	Rounded
12.5		
0.49		
-2.5		
-0.6		

3

PART 3

The range of int

An int lives between two limits



An int is stored in a finite amount of memory (4 bytes), so it must be in the range from **Integer.MIN_VALUE** to **Integer.MAX_VALUE**, inclusive.

```
System.out.println(Integer.MAX_VALUE);  
System.out.println(Integer.MIN_VALUE);
```

OUTPUT

```
2147483647  
-2147483648
```

ABOUT

2.1 billion at the top,
-2.1 billion at the bottom.

A 10-digit phone number like
5551234567 is too big to be an int.

Seventy years, in seconds



If an expression would give an int outside the range, an **integer overflow** occurs. The result is an int in the range, but not the value you expected.

```
int seconds = 70 * 365 * 24 * 60 * 60;  
System.out.println(seconds);
```

OUTPUT

-2087447296

A negative number of seconds? The true answer, 2,207,520,000, does not fit.

LIKE AN ODOMETER

Past the top, an int wraps around to the bottom of the range. No error, no crash: the program keeps running with a wrong value.

Overflow is silent, in both directions



Overflow is **not** an error or an exception. Java quietly hands you a wrong int, so it is up to you to keep values in range.

ONE BELOW THE BOTTOM

```
int low = Integer.MIN_VALUE;  
int below = low - 1;  
System.out.println(below);
```

OUTPUT

2147483647

A LITERAL TOO BIG

```
int phone = 5551234567;
```

COMPILER SAYS

integer number too large

The compiler can see a literal. It cannot see what a calculation will give.

A large, bold, teal number '4' is centered within a white rounded square that has a slight 3D effect and a shadow.

4

PART 4

Round-off error

A double keeps only so many digits



If a result is more precise than a double can store, a **round-off error** occurs: the result is rounded. To avoid it, use int values.

DOLLARS AS DOUBLE

```
System.out.println(1.1 + 2.2);
```

OUTPUT

```
3.300000000000000003
```

CENTS AS INT

```
int cents = 110 + 220;  
System.out.println(cents);
```

OUTPUT

```
330
```

That is why programs that handle money often count in cents.

int or double?

PAUSE AND TRY

Which type fits each value best? Be ready to say why.

1 Laps around a track

2 A body temperature, like 98.6

3 A store total, exact to the cent

4 A 10-digit phone number

Words to know

Widening

Java's automatic conversion of an int to a double.

Cast

(int) or (double) in front of a value, to convert it.

Truncate

Drop the digits after the decimal point.

Integer.MAX_VALUE

The largest int: 2147483647. MIN_VALUE is the smallest.

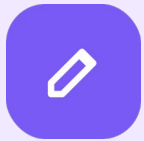
Integer overflow

An int result past the limits wraps to an unexpected int.

Round-off error

A double result rounded because it has too many digits.

Chop, round, or wrap?



Explain to your partner why these give different answers:

`(int) 4.9` and `(int) (4.9 + 0.5)`

SENTENCE STARTER

A cast to int _____ the decimal part, so to round I first _____, then _____.

Next: the Cast, Chop, and Wrap puzzle worksheet.