

Constants

Quantity	Symbol	Value in our universe
Speed of light	c	$299792458 \text{ m s}^{-1}$
Gravitational constant	G	$6.673 \times 10^{-11} \text{ m}^3 \text{ kg}^{-1} \text{ s}^{-2}$
(Reduced) Planck constant	\hbar	$1.05457148 \times 10^{-34} \text{ m}^2 \text{ kg s}^{-2}$
Planck mass-energy	$m_{\text{Pl}} = \sqrt{\hbar c / G}$	$1.2209 \times 10^{22} \text{ MeV}$
Mass of electron; proton; neutron	$m_e; m_p; m_n$	0.511; 938.3; 939.6 MeV
Mass of up; down; strange quark	$m_u; m_d; m_s$	(Approx.) 2.4; 4.8; 104 MeV
Ratio of electron to proton mass	β	$(1836.15)^{-1}$

Constants

A constant is a variable that you do not want to change (accidentally). This could be a real world constant or some kind of setting in your program.

```
final double PI = 3.14159;
```

```
final String VERSION = "v2.1";
```

Constants

The word `final` turns a variable into a constant. Typically, constants are written in `ALL_CAPITAL_LETTERS` using underscores between words.

```
final int SECONDS_PER_HOUR = 3600;
```

```
final int MAX_PLAYERS = 16;
```

```
final double DOLLARS_TO_EUROS = 0.89;
```

Constants

Eclipse will give you an error if you are trying to change the value of a constant (this is a good thing).

```
final int NUMBER_OF_PLAYERS = 16;  
  
NUMBER_OF_PLAYERS = 5;
```

Constants

Use constants instead of “magic numbers” - it will be easier to change the settings of your program later.

No!

```
damage = level * 10.0;  
gold = exp * 10.0;
```

Yes!

```
final double BASE_DAMAGE = 10.0;  
  
damage = level * BASE_DAMAGE;  
gold = exp * BASE_DAMAGE;
```

Comments

// Slides made by Carmine Guida

Comments

A comment is text that you add to your program to help (yourself) and others understand what your program does.

Comments are for humans to read.

Comments are ignored by the compiler (they do not end up in your final program anywhere).

Comments

There are two common types of comments: single-line and multi-line.

`// A single line comment begins with two slashes`

`/*`

A multi line comment begins with slash and asterisk
and does not end until it finds
an asterisk and then a slash.

`*/`

Comments

Comments can be on a line by themselves, or after a statement.

```
/*  
    Coded by: Carmine Guida  
    Email: cguida@pace.edu  
*/
```

```
area = PI * (r * r); // Area of a circle
```

Comments

Use comments to explain your thought process or what you are up to.
Do not use comments to teach someone how to program!

No!

```
total = total + 1;           // Add 1 to total
average = score / total;    // Divide score by total
```

Yes!

```
/*
    Calculate the average based on the total score and the
    number of tests taken.
*/
total = total + 1;
average = score / total;
```

Whitespace and Style

Whitespace and Style

Whitespace is the spaces, tabs and blank lines between statements.

No!

```
total=total+1;
```

```
int w=5;  
int    length = 10;
```

Yes!

```
total = total + 1;
```

```
int w = 5;  
int length = 10;
```

// Or something like this:

```
int w      = 5;  
int length = 10;
```

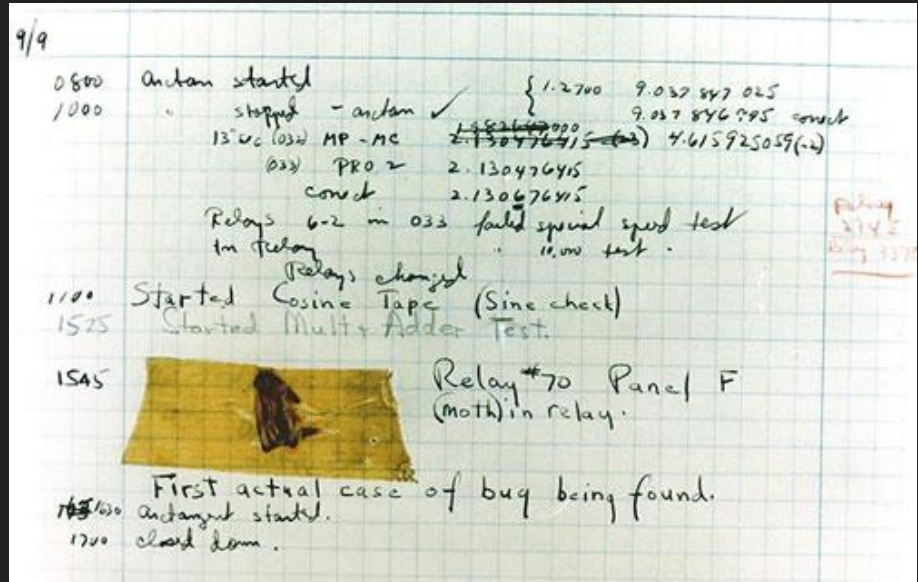
Debugging



Debugging

While the term **bug** and **debugging** were used before computers existed, engineers working on a computer in Harvard in 1947 found a moth that was causing issues with a component in the system. The moth was taped into a logbook with a comment:

“First actual case of bug being found.”



Debugging

There are a couple of different methods for debugging and troubleshooting problems with your code.

```
// Print the value so far (and remove this later)
System.out.println("average = " + average);
```

```
// Force to a specific value (and remove later)
average = 90.5;
```

Use the `integrated debugger` (coming up in the demo).

Let's Code

Don't Forget!

Check the syllabus / schedule for reading assignments and **due dates!**

Type Conversions

```
int gpa = (int)average;
```

Type Conversions

You may need to convert from one data type (such as double) to another data type (such as int). We've seen **implicit** type conversions.

```
int x = 5;
```

```
// An (int / double) is converted into a double  
System.out.println(x / 2.0);
```

Type Casting

A type cast **explicitly** converts from one datatype to another.

```
double total;
```

```
System.out.print("Total: ");  
total = scan.nextDouble();
```

```
int x = (int)total;
```

Type Casting

You may want to use type casting to make sure a formula is behaving as expected.

```
int numberOfScores = 3;  
double average = totalPoints / (double)numberOfScores;  
int distance = (int)(totalSeconds * metersPerSecond);
```

Type Casting

When in doubt, add a type cast to be sure!

```
int x = 5;  
int y = 2;
```

```
// No! z will be 2.0 (not 2.5)  
double z = x / y;
```

```
// Yes! z will be 2.5  
double z = (double)x / (double)y;
```

Overflow

$1000000 * 1000000 = -727379968$

Overflow

When storing (or printing values) some interesting things can happen when you try to go beyond the limits of the data type. For instance, `int` has a range of `-2,147,483,648` to `2,147,483,647`

```
int x = 2147483647;
```

```
System.out.println(x);           // 2147483647
System.out.println(x + 1);       // -2147483648
```

Overflow

Eclipse is aware of the ranges of data types and will present an error if you are trying to initialize to a value that is out of range. However, it does not know about the math you are doing.

```
int x = 2147483648; // Shows an error here
int y = 2147483647;

y = y + 1; // Does not know this is bad
```


Numeric Data Types

byte, short, int, long, float, double

Numeric Data Types

So far we have used `int` and `double`. There are some other numeric data types to know about.

<code>byte</code>	8 bits	-128 to 127
<code>short</code>	16 bits	-32,768 to 32,767
<code>int</code>	32 bits	-2,147,483,648 to 2,147,483,647
<code>long</code>	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Numeric Data Types

You will typically use `int` and occasionally `long` (for when you need a number greater than 2 billion). For instance as an ID column in a database or a session number for users on a very active website.

If you are working on a simple computer such as a wearable or embedded system you may need to save on memory space and use the shorter data types such as `byte` and `short`.

Numeric Data Types

Double gets its name from being a double precision float.

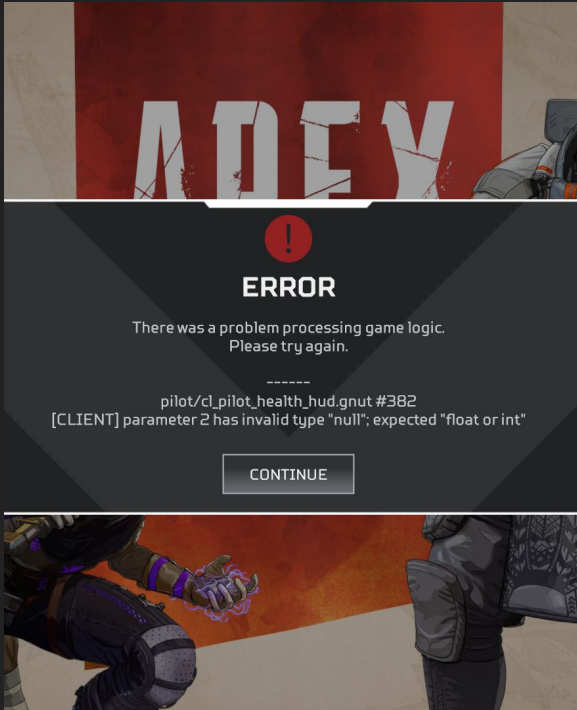
float	32 bits	-3.4×10^{38} to 3.4×10^{38}
double	64 bits	-1.7×10^{308} to 1.7×10^{308}

Numeric Data Types

`double` is the default in Java when using decimal point numbers. You will mostly use `double`.

Why not use it all the time?

Many game engines will use the `float` data type for decimal numbers. Graphics cards only started supporting 64 bit floats (FP64) around 2014. When using double precision on graphics cards performance is 1/20th then that of floats!



Let's Code

Don't Forget!

Check the syllabus / schedule for reading assignments and due dates!