Appendix: Common Errors

This appendix offers a brief overview of common errors that occur in *Processing*, what those errors mean and why they occur. The language of error messages can often be cryptic and confusing, mostly due to their origins in the Java programming language.

For these error messages, we will use the following conventions:

Variables in the error messages will be named "myVar".
Arrays in the error messages will be named "myArray".
If the error is specifically related to an object variable, the variable will be named "myObject".
If the error is related to a class, the class will be named "Thing".
If the error is related to a function, the function will be named "function".

ERROR	PAGE NO.
unexpected token: _____	440
Found one too many { characters without a } to match it.	440
No accessible field named "myVar" was found in type "Temporary_####_####"	441
The variable "myVar" may be accessed here before having been definitely assigned a value.	442
java.lang.NullPointerException	442
Type "Thing" was not found.	444
Perhaps you wanted the overloaded version "type function (type $1, type $2, …);" instead?	445
No method named "fnction" was found in type "Temporary_####_####". However, there is an accessible method "function" whose name closely matches the name "fnction".	445
No accessible method with signature "function (type, type, …)" was found in type "Temporary_3018_2848".	445
java.lang.ArrayIndexOutOfBoundsException	446

Error:

unexpected token: something

Translation:

I'm expecting a piece of code here that you forgot, most likely some punctuation, like a semicolon, parenthesis, curly bracket, and so on.

This error is pretty much always caused by a typo. Unfortunately, the line of code that the error points to can be misleading. The "something" is sometimes correct and the actual error is caused by line before or after. Any time you forget or include an improper comma, semicolon, quote, parenthesis, bracket, and so on, you will get this error. Here are some examples:

ERROR	CORRECTED
`int val = 5` Missing semicolon.	`int val = 5;`

`if (x < 5 {`
`  ellipse(0,0,10,10);`
`}`
Missing close parenthesis.
	`if (x < 5) {`
	`  ellipse(0,0,10,10);`
	`}`

`for (int i = 0, i < 5, i++) {`
`  println(i);`
`}`
For loop should be separated by semicolons, not commans.
	`for(inti = 0;i < 5;i++)`
	`  println(i);{`

Error:

Found one too many { characters without a } to match it.

Translation:

You forgot to close a block of code, such as an if statement, a loop, a function, a class, and so on.

Any time you have an open curly bracket in your code ("{"), you must have a matching close curly bracket ("}"). And since blocks of code are often nested inside each other, it is easy to forget one by accident, resulting in this error.

ERROR	CORRECTED

`void setup() {`
`  for (int i = 0; i < 10; i++) {`
`  if (i > 5) {`
`  line(0,i,i,0);`
`  println("i is greater than 5");`
`  }`
`}`
Missing a close curly bracket!
	`void setup() {`
	`  for (int i = 0; i < 10; i++) {`
	`  if (i > 5) {`
	`  line(0,i,i,0);`
	`  println("i is greater than 5");`
	`  }`
	`  }`
	`}`

Error:

No accessible field named "myVar" was found in type "Temporary_7665_1082"

Translation:

I do not know about the variable "myVar." It is not declared anywhere I can see.

This error will happen if you do not declare a variable. Remember, you can't use a variable until you have said what type it should be. You will get the error if you write:

```
myVar = 10;
```

instead of:

```
int myVar = 10;
```

Of course, you should only declare the variable once, otherwise you can run into trouble. So this is perfectly OK:

```
int myVar = 10;
myVar = 20;
```

OK! The variable was declared.

This error can also occur if you try to use a local variable outside of the block of code where it was declared. For example:

```
if (mousePressed) {
  int myVar = 10;
}
ellipse(myVar,10,10,10);
```

ERROR! myVar is local to the if statement so it cannot be accessed here.

Here is a corrected version:

```
int myVar = 0;
if (mousePressed) {
  myVar = 10;
}
ellipse(myVar,10,10,10);
```

OK! The variable is declared outside of the if statement.

Or if you have declared a variable in ***setup()***, but try to use it in ***draw()***.

```
void setup() {
  int myVar = 10;
}

void draw() {
  ellipse(myVar,10,10,10);
}
```

ERROR! myVar is local to the ***setup()*** so it cannot be accessed here.

Corrected:

```
int myVar = 0;

void setup() {
  myVar = 10;
}
```

```
void draw() {
  ellipse(myVar,10,10,10);
}
```

OK! The variable is global.

The same error will occur if you use an array in any of the above scenarios.

```
myArray[0] = 10;
```

ERROR! No array was declared.

Error:

The variable "myVar" may be accessed here before having been definitely assigned a value.

Translation:

You declared the variable "myVar", but you did not initialize it. You should give it a value first.

This is a pretty simple error to fix. Most likely, you just forgot to give your variable its default starting value. This error only occurs for local variables (with global variables *Processing* will either not care, and assume the value is zero, or else throw a *NullPointerException*).

```
int myVar;
line(0, myVar,0,0);
```

ERROR! myVar does not have a value.

```
int myVar = 10;
line(0, myVar,0,0);
```

OK! myVar equals ten.

This error can also occur if you try to use an array before allocating its size properly.

```
int[] myArray;
myArray[0] = 10;
```

ERROR! myArray was not created properly.

```
int[] myArray = new int[3];
myArray[0] = 10;
```

OK! myArray is an array of three integers.

Error:

java.lang.NullPointerException

Translation:

I have encountered a variable whose value is null. I can't work with this.

The ***NullPointerExceptio***n error can be one of the most difficult errors to fix. It is most commonly caused by forgetting to initialize an object. As mentioned in Chapter 8, when you declare a variable that is an object, it is first given the value of ***null***, meaning nothing. (This does not apply to primitives, such as integers and floats.) If you attempt to then use that variable without having initialized it (by calling the Constructor), this error will occur. Here are a few examples (that assume the existence of a class called "Thing").

```
Thing thing;
void setup() {

}
```

```
void draw() {
  thing.display();
}
```

ERROR! "thing" was never initialized and therefore has the value null.

Corrected:

```
Thing thing;
void setup() {
  thing = new Thing();
}

void draw() {
  thing.display();
}
```

OK! "thing" is not null because it was initialized with the constructor.

Sometimes, the same error can occur if you do initialize the object, but as a local variable by accident.

```
Thing thing;
void setup() {
  Thing thing = new Thing();
}

void draw() {
  thing.display();
}
```

ERROR! This line of code declares and initializes a different "thing" variable (even though it has the same name). It is a local variable for only ***setup()***. The global "thing" is still null!

The same goes for an array as well. If you forget to initialize the elements, you will get this error.

```
Thing[] things = new Thing[10];
for (int i = 0; i < things.length; i++) {
  things[i].display();
}
```

ERROR! All the elements on the array are null!

Corrected:

```
Thing[] things = new Thing[10];
for (int i = 0; i < things.length; i++) {
  things[i] = new Thing();
}

for (int i = 0; i < things.length; i++) {
  things[i].display();
}
```

OK! The first loop initialized the elements of the array.

Finally, if you forget to allocate space for an array, you can also end up with this error.

```
int[] myArray;

void setup() {
  myArray[0] = 5;
}
```

ERROR! myArray is null because you never created it and gave it a size.

Corrected:

```
int[] myArray = new int[3];
void setup() {
  myArray[0] = 5;
}
```

OK! myArray is an array of three integers.

Error:

Type "Thing" was not found.

Translation:

You are trying to declare a variable of type Thing, but I do not know what a Thing is.

This error occurs because you either (a) forgot to define a class called "Thing" or (b) made a typo in the variable type.

A typo is pretty common:

```
intt myVar = 10;
```

ERROR! You probably meant to type "int" not "intt."

Or maybe you want to create an object of type Thing, but forgot to define the thing class.

```
Thing myThing = new Thing();
```

ERROR! If you did not define a class called "Thing."

This will work, of course, if you write:

```
void setup() {
  Thing myThing = new Thing();
}

class Thing {
  Thing() {
  }
}
```

OK! You did declare a class called "Thing."

Finally, the error will also occur if you try to use an object from a library, but forget to import the library.

```
void setup() {
  Capture video = new Capture(this,320,240,30);
}
```

ERROR! You forgot to import the video library.

Corrected:

```
import processing.video.*;
void setup() {
  Capture video = new Capture(this,320,240,30);
}
```

OK! You imported the library.

Error:

Perhaps you wanted the overloaded version "type function (type $1, type $2, …);" instead?

Translation:

You are calling a function incorrectly, but I think I know what you mean to say. Did you want call this function with these arguments?

This error occurs most often when you call a function with the incorrect number of arguments. For example, to draw an ellipse, you need an *x* location, *y* location, width, and height. But if you write:

```
ellipse(100,100,50);
```

ERROR! ***ellipse()*** requires four arguments.

You will get the error: "Perhaps you wanted the overloaded version 'void ellipse(float $1, float $2, float $3, float $4);' instead?" The error lists the function definition for you, indicating you need four arguments, all of floating point. The same error will also occur if you have the right number of arguments, but the wrong type.

```
ellipse(100,100,50,"Wrong Type of Argument");
```

ERROR! ***ellipse()*** can not take a String!

Error:

No method named "fnction" was found in type "Temporar y_9938_7597". However, there is an accessible method "function" whose name closely matches the name "fnction".

Translation:

You are calling a function I have never heard of, however, I think I know what you want to say since there is a function I have heard of that is really similar.

This is a similar error and pretty much only happens when you make a typo in a function's name.

```
elipse(100,100,50,50);
```

ERROR! You have the right number of arguments, but you spelled "ellipse" incorrectly.

Error:

No accessible method with signature "function (type, type, …)" was found in type "Temporary_3018_2848".

Translation:

You are calling a function that I have never heard of. I have no idea what you are talking about!

This error occurs if you call a function that just plain does not exist, and *Processing* can't make any guess as to what you meant to say.

```
functionCompletelyMadeUp(200);
```

ERROR! Unless you have defined this function, *Processing* has no way of knowing what it is.

Same goes for a function that is called on an object.

```
Capture video = new Capture(this,320,240,30);

video.turnPurple();
```

ERROR! There is no function called ***turnPurple()*** in the Capture class.

Error:
java.lang.ArrayIndexOutOfBoundsException: ##

Translation:
You tried to access an element of an array that does not exist.

This error will happen when the index value of any array is invalid. For example, if your array is of length 10, the only valid indices are zero through nine. Anything less than zero or greater than nine will produce this error.

```
int[] myArray = new int[10];

myArray[-1] = 0
```

ERROR! –1 is not a valid index.

```
myArray[0] = 0;
myArray[5] = 0;
```

OK! 0 and 5 are valid indices.

```
myArray[10] = 0;
myArray[20] = 0;
```

ERROR! 10 is not a valid index.

ERROR! 20 is not a valid index.

This error can be a bit harder to debug when you are using a variable for the index.

```
int[] myArray = new int[100];
myArray[mouseX] = 0;
```

ERROR! ***mouseX*** could be bigger than 99.

```
int index = constrain(mouseX,0,myArray.length-1);

myArray[index] = 0;
```

OK! ***mouseX*** is constrained between 0 and 99 first.

A loop can also be the source of the problem.

```
for (int i = 0; i < 200; i++) {
  myArray[i] = 0;
}
```

ERROR! *i* loops past 99.

```
for (int i = 0; i < myArray.length; i++) {
  myArray[i] = 0;
}
```

OK! Using the length property of the array as the exit condition for the loop.

```
for (int i = 0; i < 200; i++) {
  if (i < myArray.length) {
  myArray[i] = 0;
  }
}
```

OK! If your loop really needs to go to 200, then you could build in an if statement inside the loop.