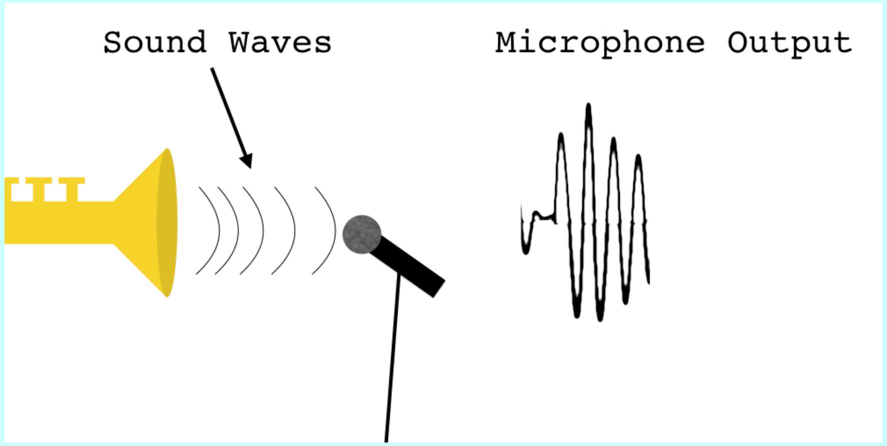
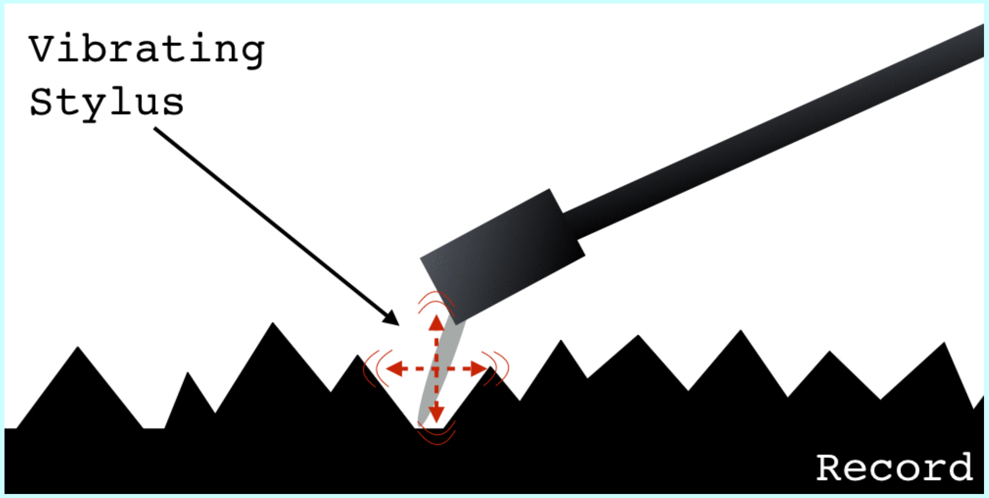
# Simple Arrays

**Short introduction to processing data**

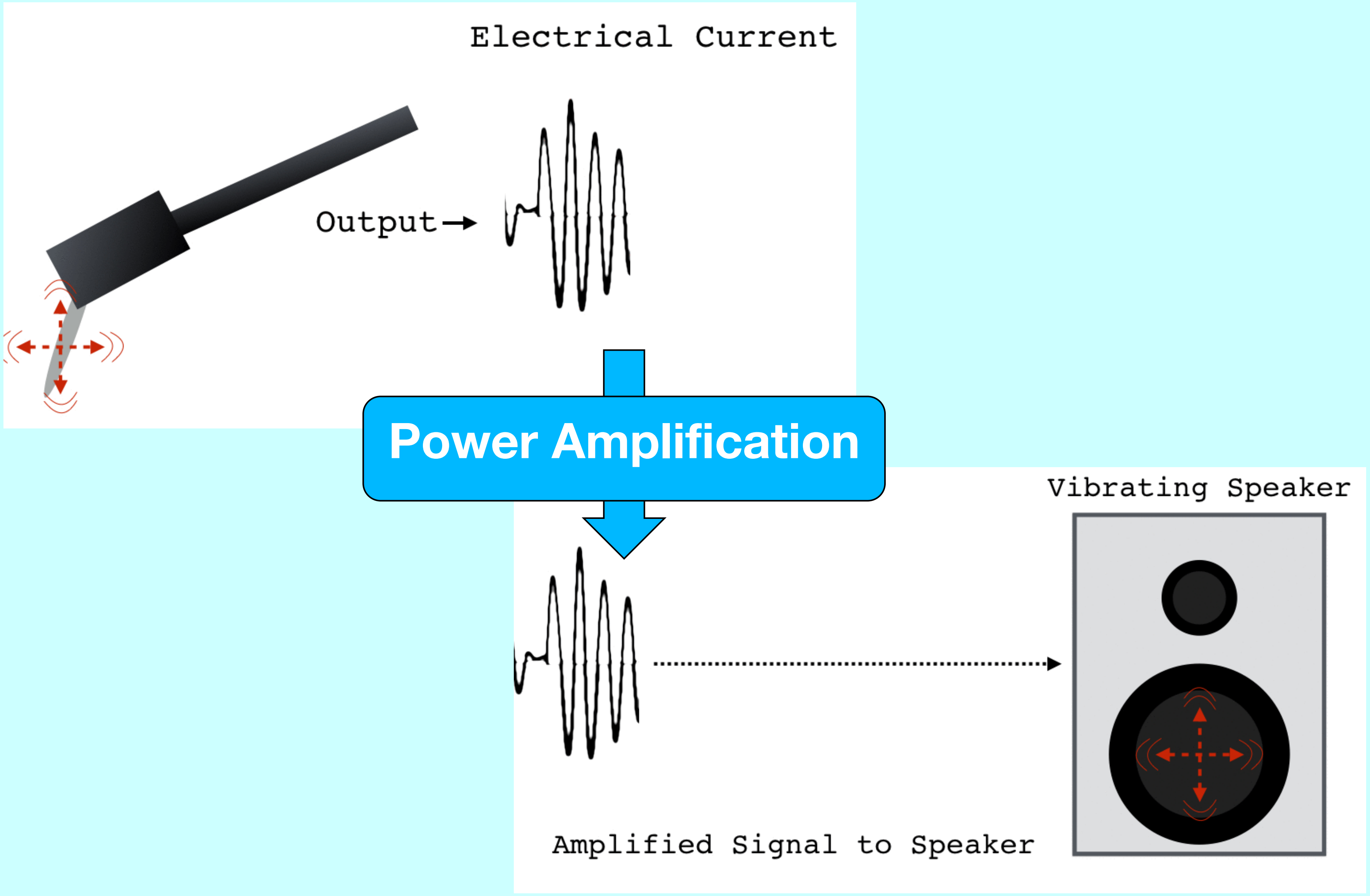
# Recording and storing sound
## *Analog*



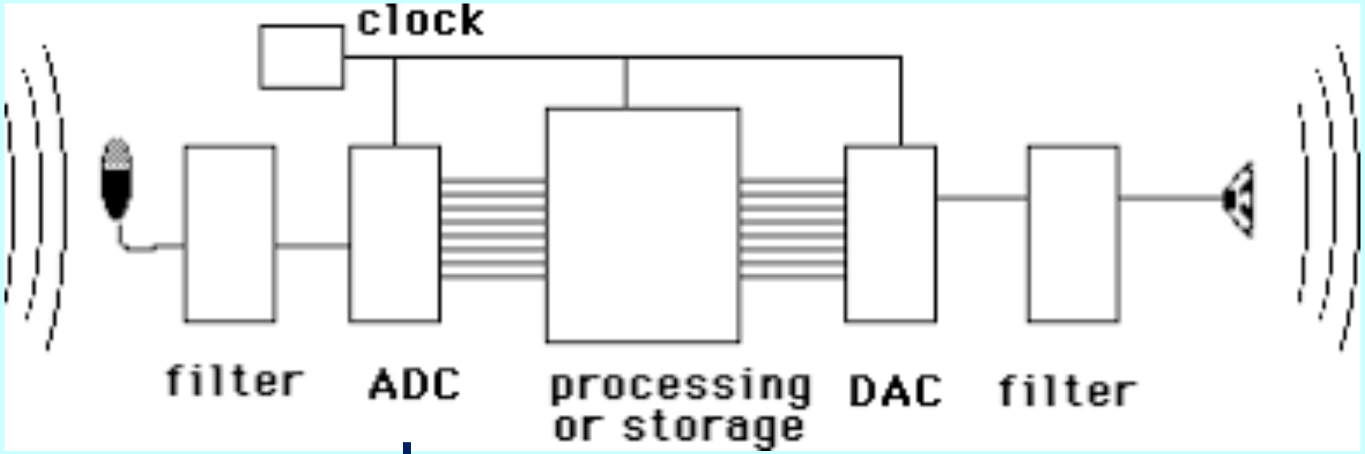


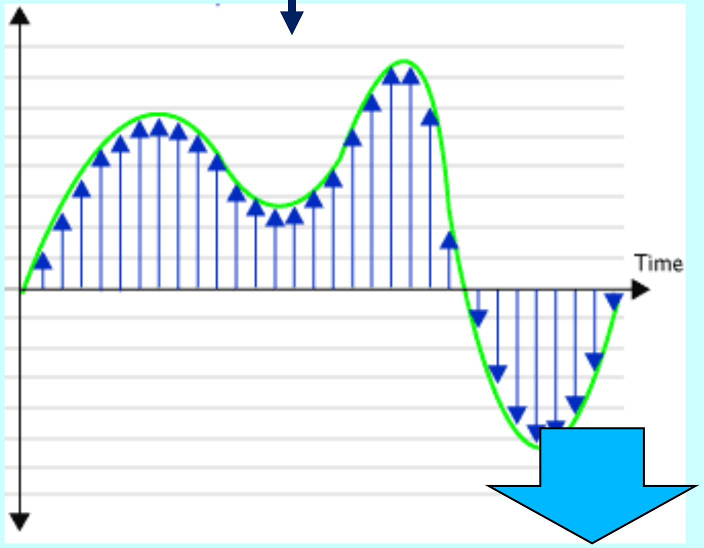
http://www.soundsetal.com/blog-how-do-vinyl-records-work/

# Playback of stored analog sound



# What about digital?





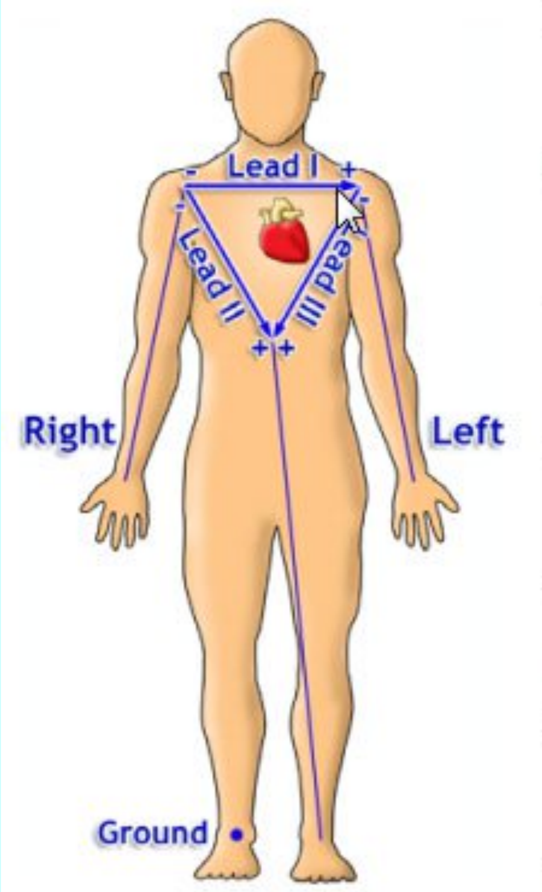
**Sampling:**
***Audio CD quality***
44100 samples/second

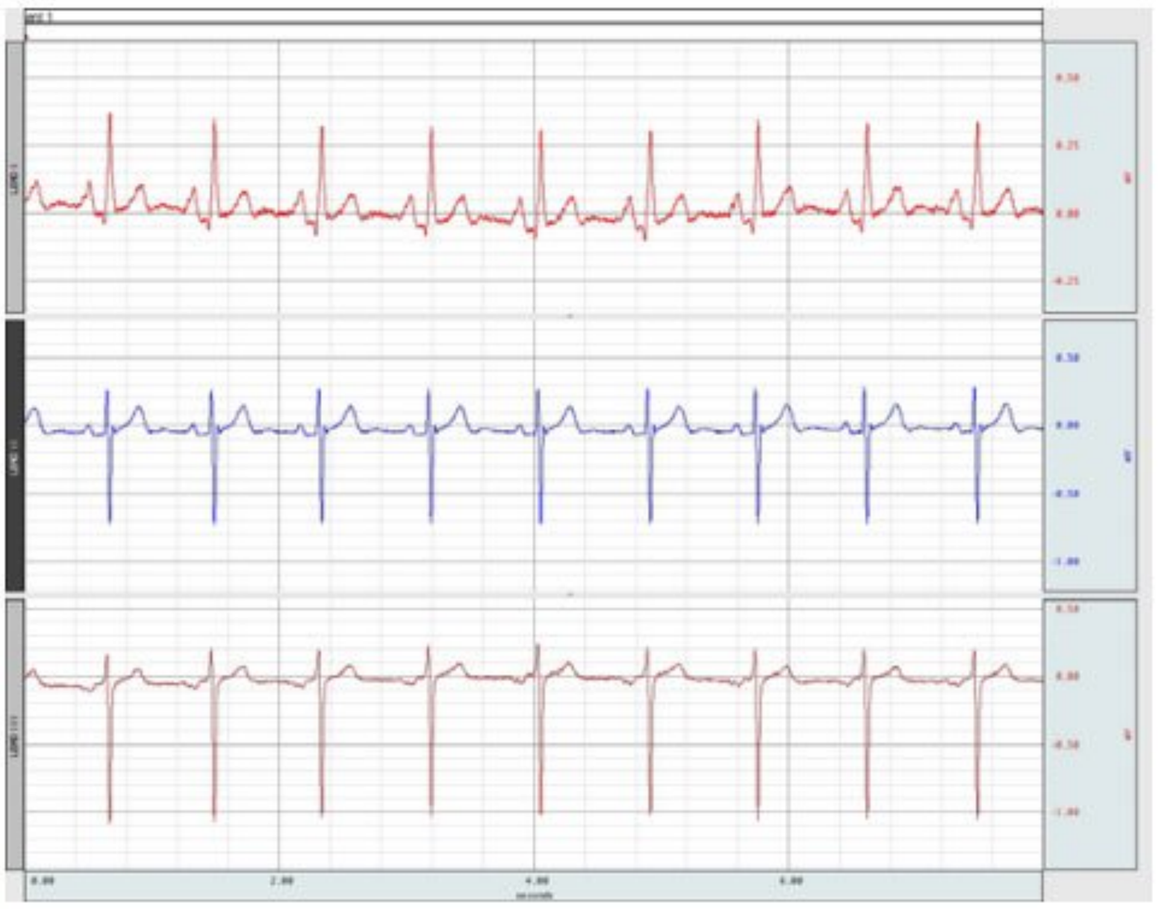
**Quantization:**
representing real values with fixed precision:
***Audio CD quality***
$2^{16}$ possible distinct values

Array

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 15 | 32 | 38 | 42 | 41 | 40 | 37 | 35 | 27 | … |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | … |

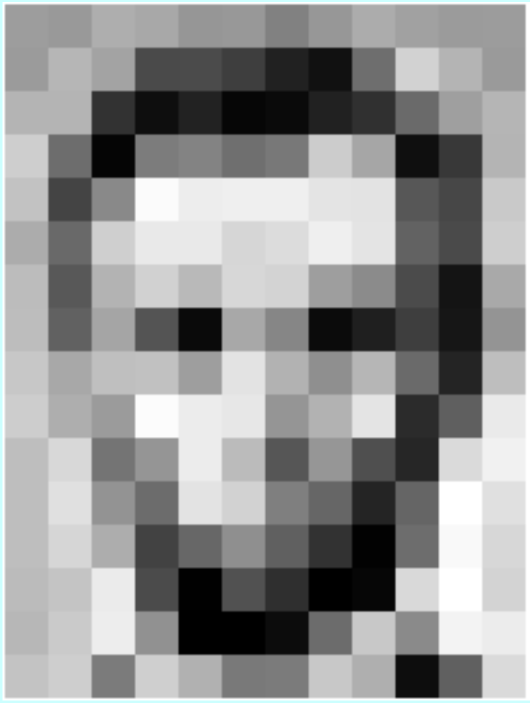
# Biological signals

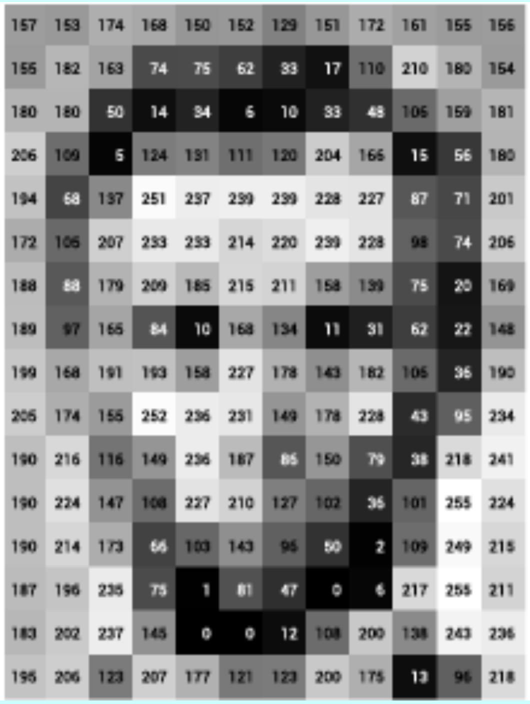




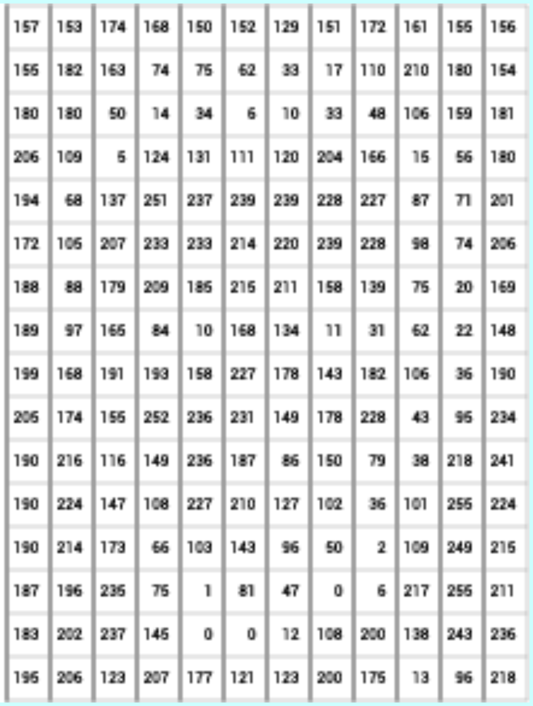
In computer memory, represented as array of data points/measurements

# Representing images





| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |



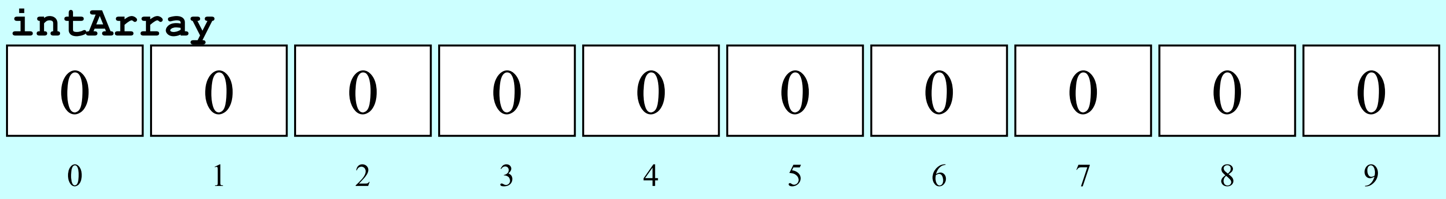
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

**! The order of data is as important as the values themselves**

# Declaring an Array Variable

```
type[] name = new type[n];
```

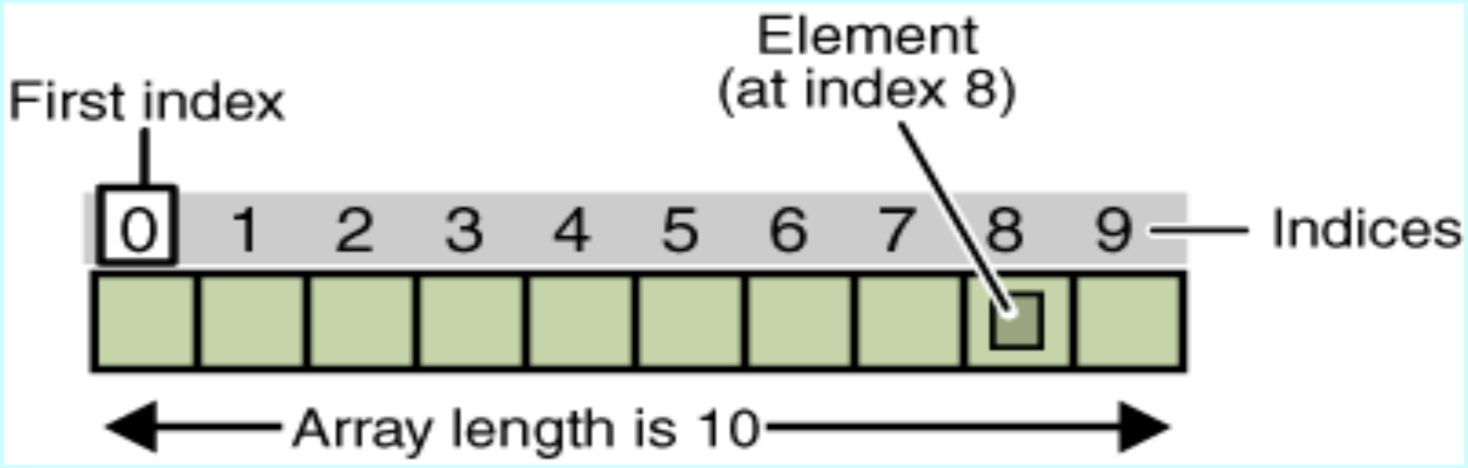
```
int[] intArray = new int[10];
```



## Arrays: Basic properties

1. *An array is ordered.(considering indexes not the contents)*

2. *An array is homogeneous.*



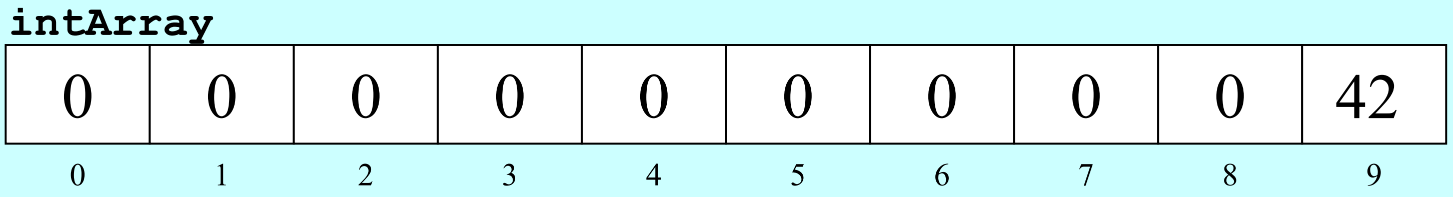
## Array Selection

- You can, for example, select the initial element by writing

```
intArray[0]
```

- Assigning a value to an element

```
intArray[9] = 42;
```



## Cycling through Array Elements

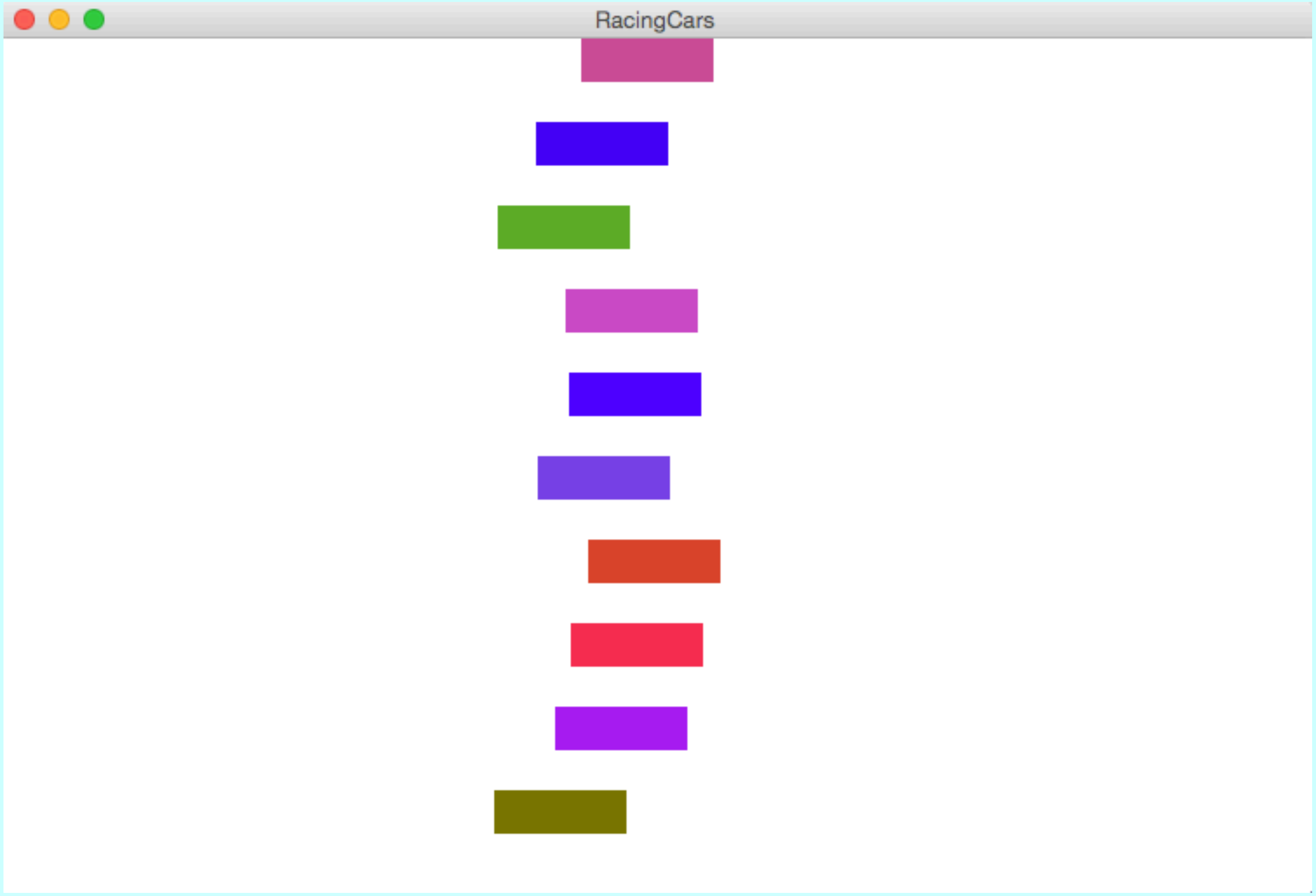
- Cycling through each of the array elements

```
for (int i = 0; i < array.length; i++) {
    Operations involving the ith element of the array
}
```

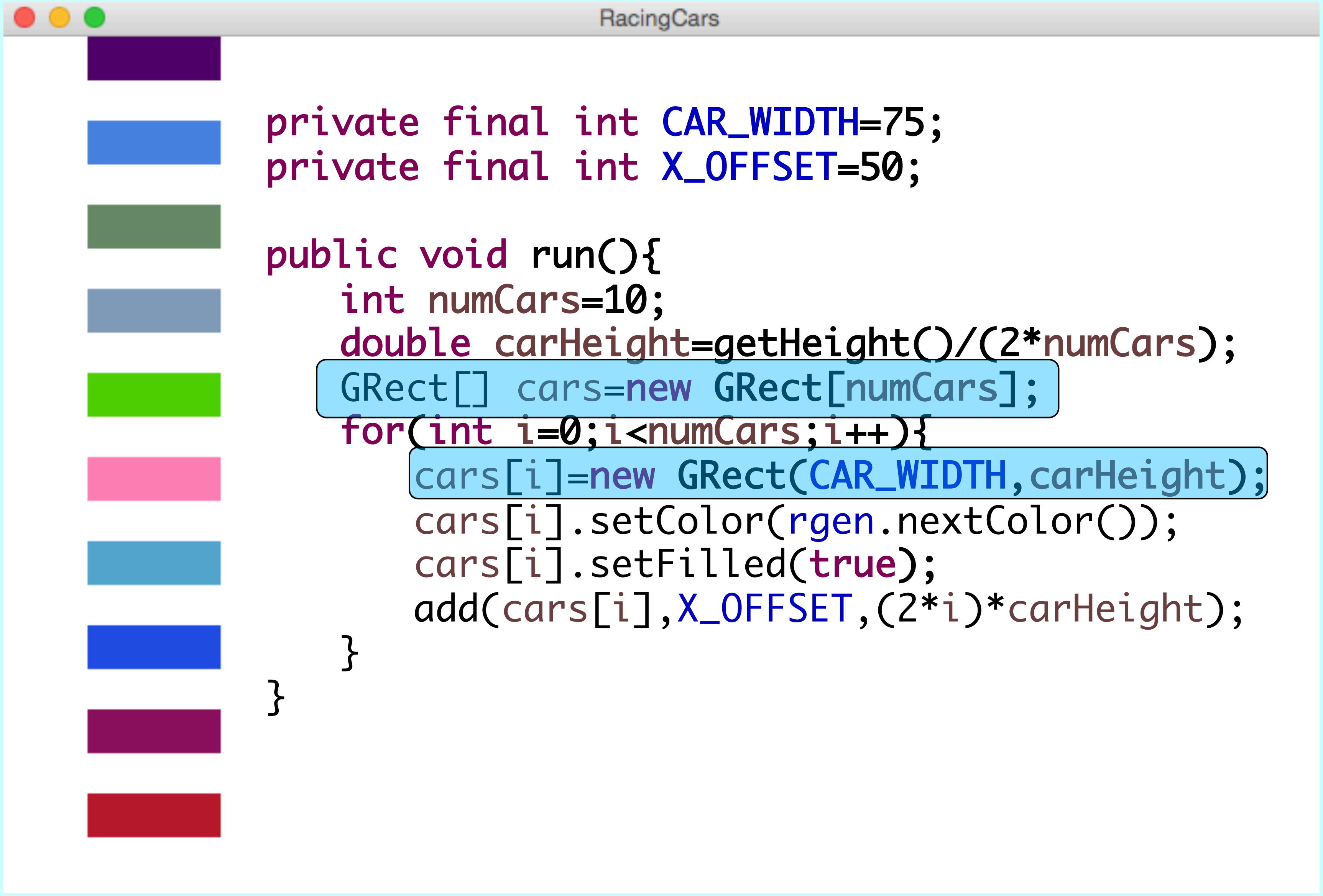
- As an example, you can reset every element in **`intArray`** to -1 using the following **`for`** loop:

```
for (int i = 0; i < intArray.length; i++) {
    intArray[i] = -1;
}
```

# An array of graphical objects



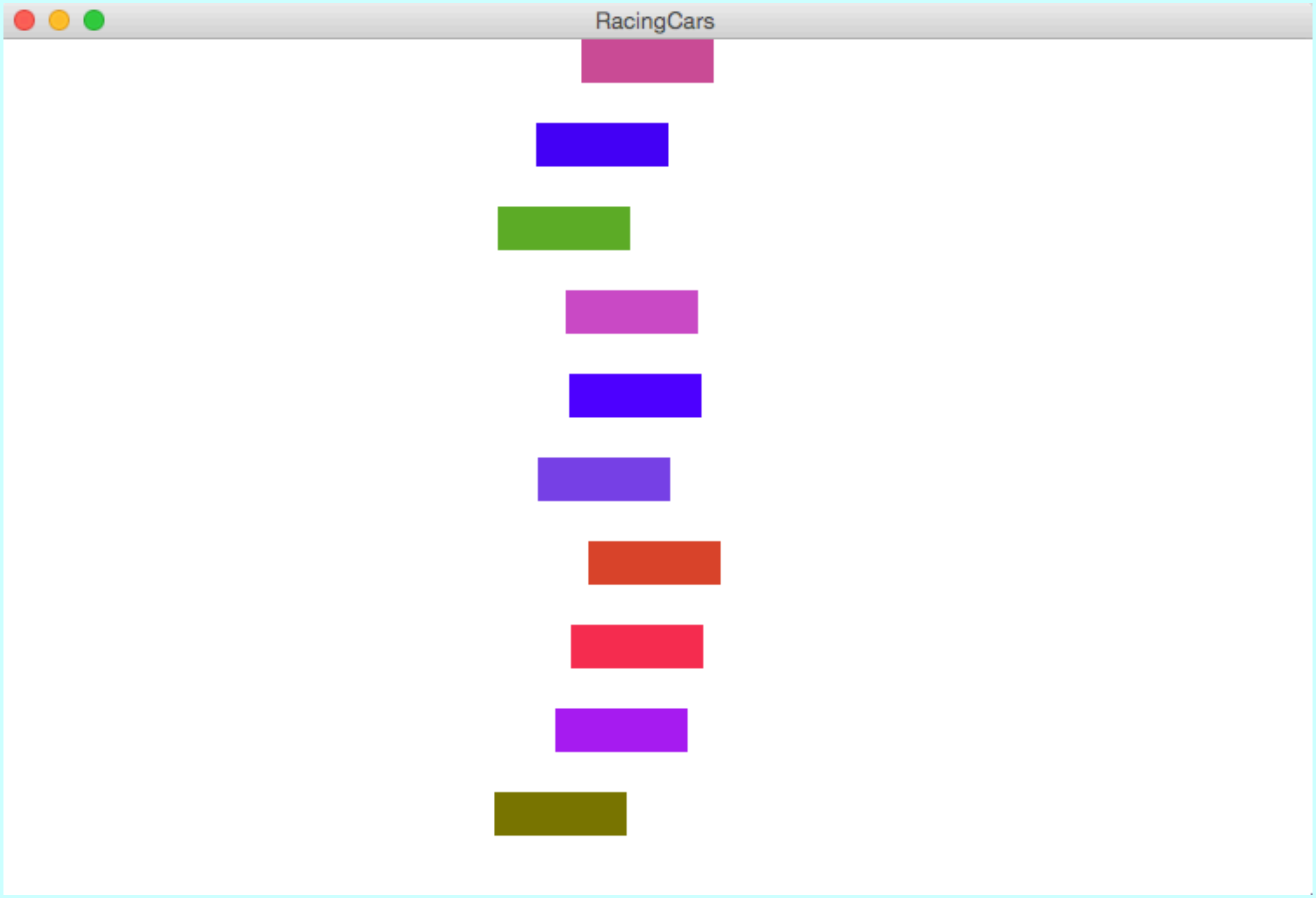
# Let’s start by placing an array of cars



```
private final int CAR_WIDTH=75;
private final int X_OFFSET=50;

public void run(){
    int numCars=10;
    double carHeight=getHeight()/(2*numCars);
    GRect[] cars=new GRect[numCars];
    for(int i=0;i<numCars;i++){
        cars[i]=new GRect(CAR_WIDTH,carHeight);
        cars[i].setColor(rgen.nextColor());
        cars[i].setFilled(true);
        add(cars[i],X_OFFSET,(2*i)*carHeight);
    }
}
```

# Animating an array of objects



# Initializing Arrays

- Java makes it easy to initialize the elements of an array as part of a declaration. The syntax is

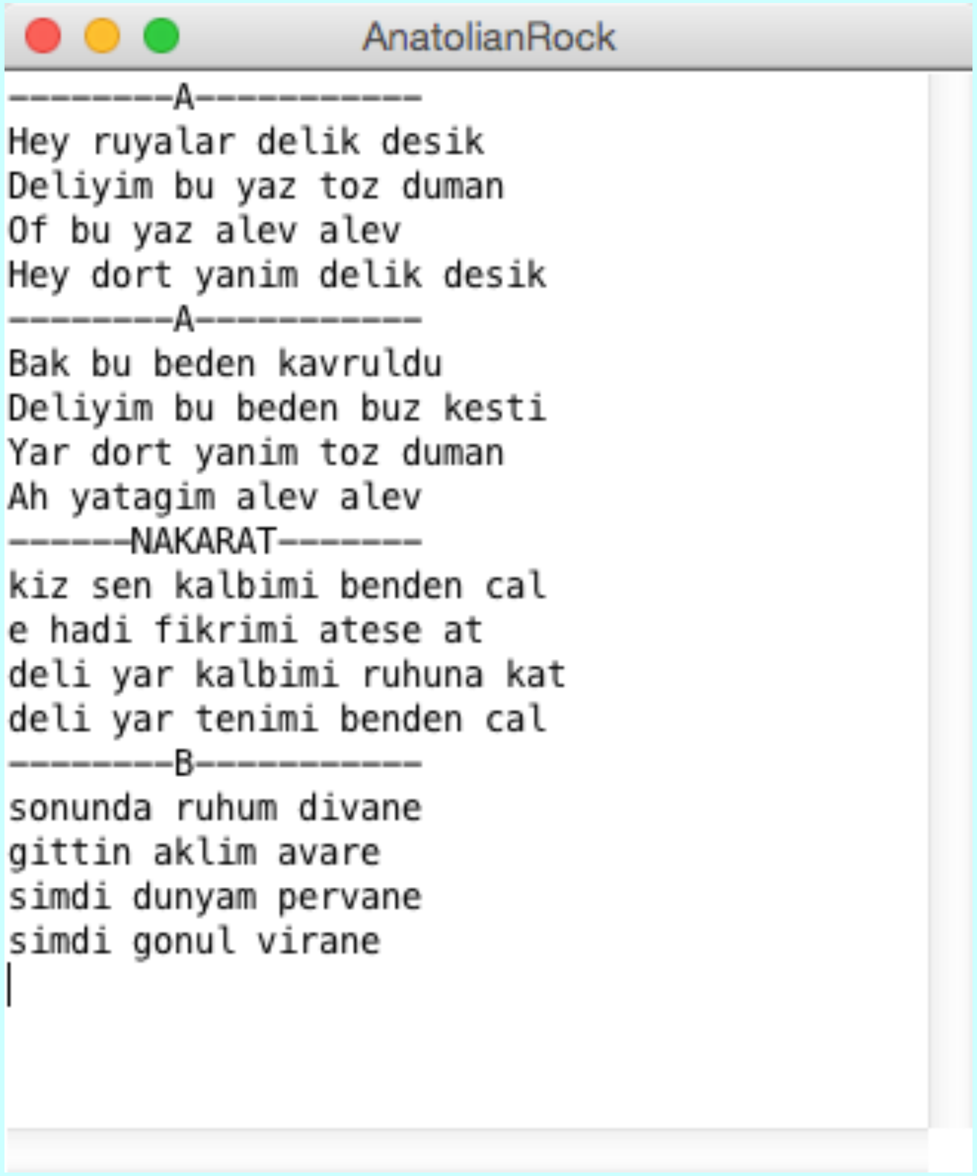
```
type[]  name  =  { elements };
```

- For example, the following declaration initializes the variable **`powersOfTen`** to the values $10^0$, $10^1$, $10^2$, $10^3$, and $10^4$:

```
int[] powersOfTen = { 1, 10, 100, 1000, 10000 };
```

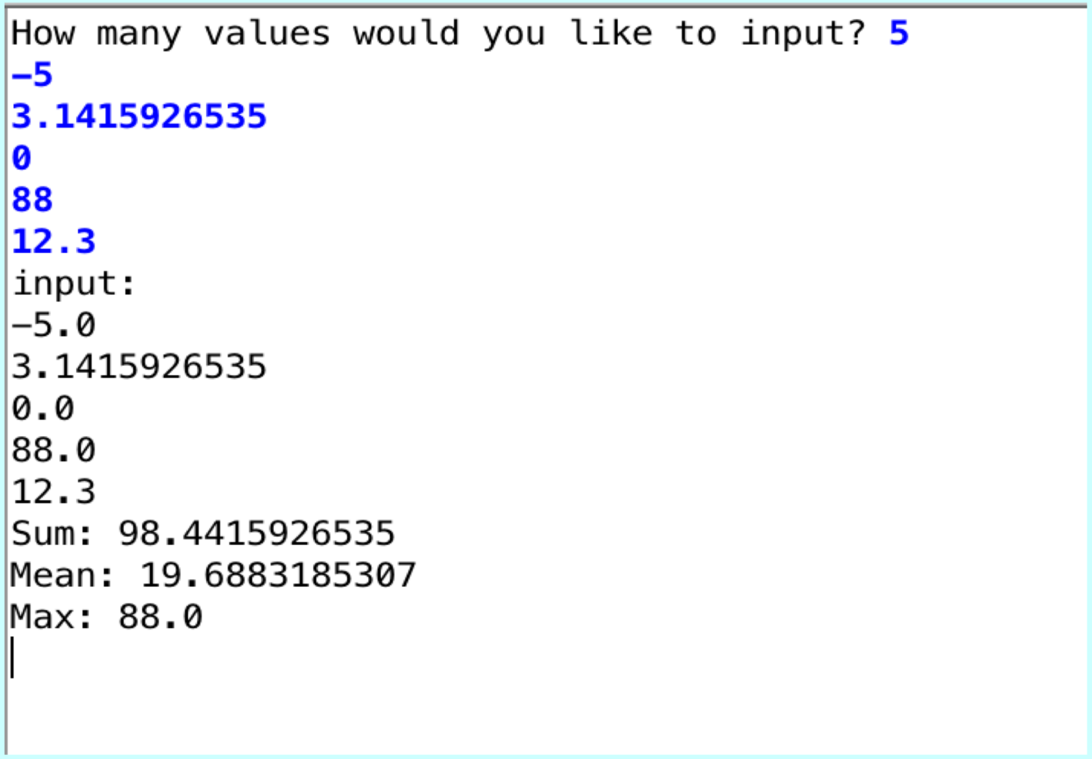
This declaration creates an integer array of length 5 and initializes the elements as specified.

# Lyrics generator: Anatolian rock



```
---------A-----------
Hey ruyalar delik desik
Deliyim bu yaz toz duman
Of bu yaz alev alev
Hey dort yanim delik desik
---------A-----------
Bak bu beden kavruldu
Deliyim bu beden buz kesti
Yar dort yanim toz duman
Ah yatagim alev alev
------NAKARAT-------
kiz sen kalbimi benden cal
e hadi fikrimi atese at
deli yar kalbimi ruhuna kat
deli yar tenimi benden cal
---------B-----------
sonunda ruhum divane
gittin aklim avare
simdi dunyam pervane
simdi gonul virane
```

# Exercise: Finding minimum, maximum and mean of an array of integers



```
How many values would you like to input? 5
-5
3.1415926535
0
88
12.3
input:
-5.0
3.1415926535
0.0
88.0
12.3
Sum: 98.4415926535
Mean: 19.6883185307
Max: 88.0
```

## Exercise: Finding minimum, maximum and mean of an array of integers

```
public void run() {
    int numValues=readInt("Number of values to be entered: ");
    /*Creating the array*/
    int[] values=new int[numValues];
    for(int i=0;i<values.length;i++) {
        values[i]=readInt("Specify input for index "+i+" :");
    }
    println("Max: "+findMax(values));
    println("Min: "+findMin(values));
    println("Mean: "+findMean(values));
}

private int findMax(int[] inputArray) {
    int max=0;
    /*Implement the method*/
    return max;
}
```

# Review: methods

```
public class MethodsReview extends ConsoleProgram{
   public void run() {

      printInfo();

   }


   private void printInfo( ) {
      println("This method prints some instructions");
      println("1-Don't use arguments ");
      println("...");

   }
}
```

You should call the method to make use of it

This method does not take any input

This method does not return any output

# Review: methods

```
public void run() {

   int x = sum2ints(5,6);

}
```

You should call the method with two int inputs

This method takes two int inputs

```
private int sum2ints(int x,int y) {

   int sum = x + y;
   return sum;

}
```

This method returns an int output

## What is the value printed?

```
public class MethodsReview extends ConsoleProgram{

   private int var1=0;

   public void run() {
      someMethod();
      println(var1);
   }

   private void someMethod(){
      int var1 = 5;
   }
}
```

## What is the value printed?

```
public class MethodsReview extends ConsoleProgram{

   private int var1=0;

   public void run() {
      someMethod();
      println(var1);
   }

   private void someMethod(){
      var1 = 5;
   }
}
```

## What is the value printed?

```
public class MethodsReview extends ConsoleProgram{

   private int var1=0;

   public void run() {
      someMethod();
      println(var1);
   }

   private int someMethod(){
      int var1 = 5;
      return 10;
   }
}
```

# What is the value printed?

```java
public class MethodsReview extends ConsoleProgram{

   private int var1=0;

   public void run() {
      var1=someMethod();
      println(var1);
   }

   private int someMethod(){
      return 10;
   }
}
```