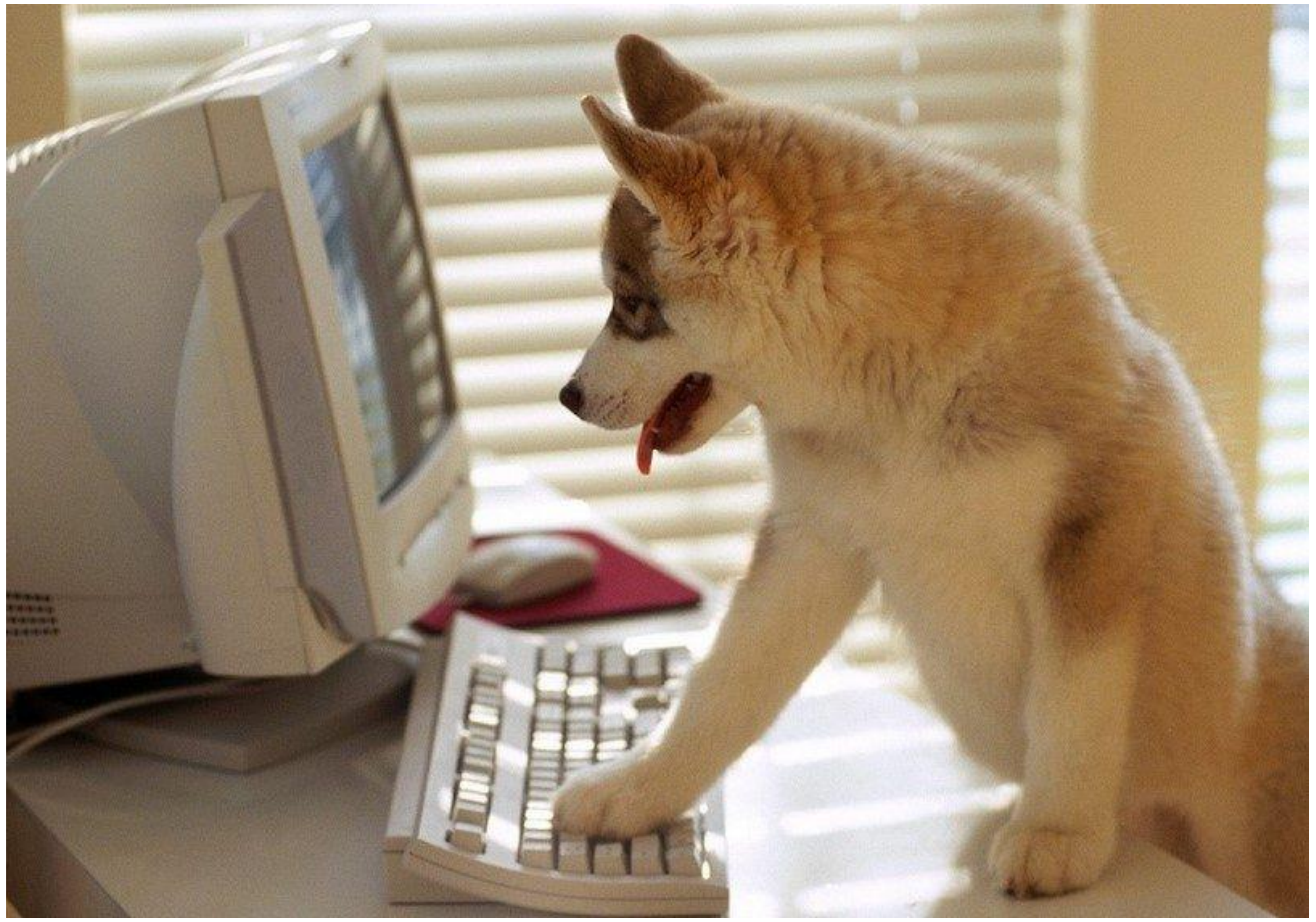
Programming Tutorial [Basics]



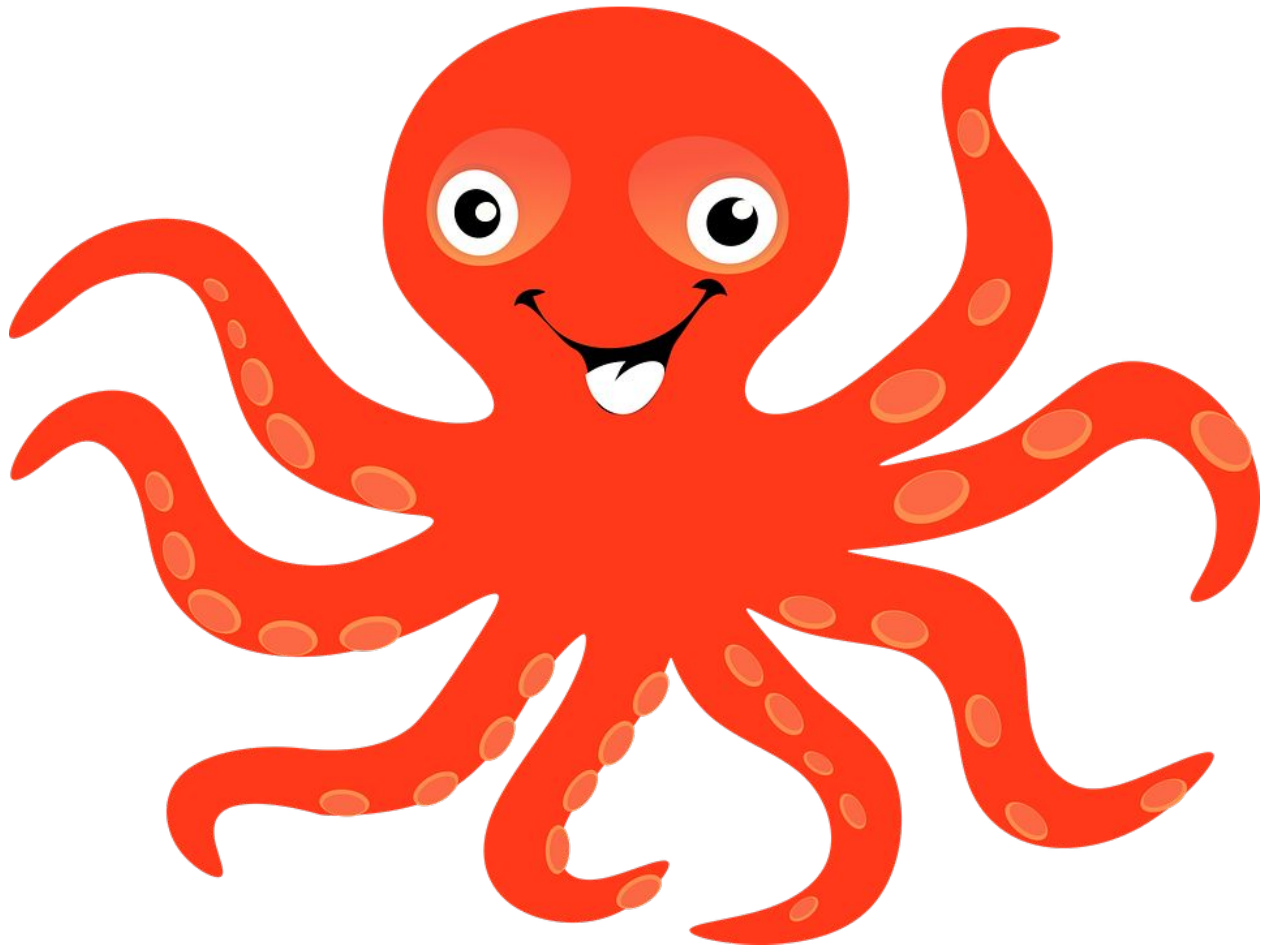
Organizational Stuff

No ECTS! (But please don’t run away!)

Threads

Why Threads?

- Parallelism leads to better performance



Threads

```
public class ThreadDemo extends Threads{

    public void run(){
        try{
            System.out.println("Thread "
                                   +Thread.
                                   currentThread().
                                   getId() +
                                   " is running.");
        }
        catch(InterruptedException e){
            e.printStackTrace();
        }
    }

}
```

```
public class RunnableDemo implements
    Runnable{

    public void run(){
        try{
            System.out.println("Thread "
                                   +Thread.
                                   currentThread().
                                   getId() +
                                   " is running.");
        }
        catch(InterruptedException e){
            e.printStackTrace();
        }
    }

}
```

```
import java.util.concurrent.Callable;

public class CallableDemo implements
    Callable<Integer>{

    private static int cntr = 0;
    private final int id = ++cntr;

    public Integer call(){
        return id;
    }

}
```

Threads

```
public class ThreadDemo extends Threads{

    public void run(){
        try{
            System.out.println("Thread "
                                   +Thread.
                                   currentThread().
                                   getId() +
                                   " is running.");
        }
        catch(InterruptedException e){
            e.printStackTrace();
        }
    }

}
```

```
public class RunnableDemo implements
    Runnable{

    public void run(){
        try{
            System.out.println("Thread "
                                   +Thread.
                                   currentThread().
                                   getId() +
                                   " is running.");
        }
        catch(InterruptedException e){
            e.printStackTrace();
        }
    }

}
```

```
import java.util.concurrent.Callable;

public class CallableDemo implements
    Callable<Integer>{

    private static int cntr = 0;
    private final int id = ++cntr;

    public Integer call(){
        return id;
    }

}
```

Threads

```
public static void main(String[] args){
    ThreadDemo td = new ThreadDemo();
    td.start();
}
```

```
public static void main(String[] args){
    RunnableDemo rd = new RunnableDemo();
    New Thread(rd).start();
}
```

```
public static void main(String[] args){
    CallableDemo cd = new CallableDemo();
    System.out.println(“id: ” + cd.call());
}
```

Threads

```
Public static void main(String[] args){
  Thread t = Thread.currentThread();
  System.out.println(“Name: ” + t.getName());
  System.out.println(“ID: ” + t.getId());
  System.out.println(“Priority: ” + t.getPriority());
  System.out.println(“State: ” + t.getState());
}
```

Threads

```
public class SimpleThread extends Thread{

    public void run(){
       for(int i = 0; i<100; i++){
          System.out.println(getState());
       }
    }

}

public class Driver{

    public static void main(String[] args){
       SimpleThread t = new SimpleThread();
       t.start();

       try{
          Thread.sleep(1000);
       }
       catch(InterruptedException e){
          e.printStackTrace();
       }
       System.out.println(t.getState());
    }

}
```

```
public class SimpleThread extends Thread{
   volatile boolean isRunning = false
    public void stopThread(){
       isRunning = false;
    }
    public void run(){
       while(running){
          System.out.println(getState());
       }
    }
}

public class Driver{
    public static void main(String[] args){
       SimpleThread t = new SimpleThread();
       t.start();

       try{
          Thread.sleep(1000);
       }
       catch(InterruptedException e){
          e.printStackTrace();
       }
       t.stopThread();
    }
}
```

Threads

Important methods:

```
sleep(int millis)
interrupt()
isInterrupted()
yield()
join()
```

Threads

Important methods:

```
sleep(int millis)
interrupt()
isInterrupted()
yield()
join()
```

Threads

```
int i1;
int geti1() {
  return i1;
}

volatile int i2;
int geti2() {
  return i2;
}

int i3;
synchronized int geti3() {
  return i3;
}
```

Threads

```
int i1;
int geti1() {
   return i1;
}

volatile int i2;
int geti2() {
   return i2;
}

int i3;
synchronized int geti3() {
   return i3;
}
```

Threads

In this course we will use Github Classroom

1. Get a Github Account if you don’t have one
2. Go to: https://classroom.github.com/a/Wj57I2Rp (or scan the QR Code with your phone)
3. Authorize Github and accept the assignment
4. Click on the repository