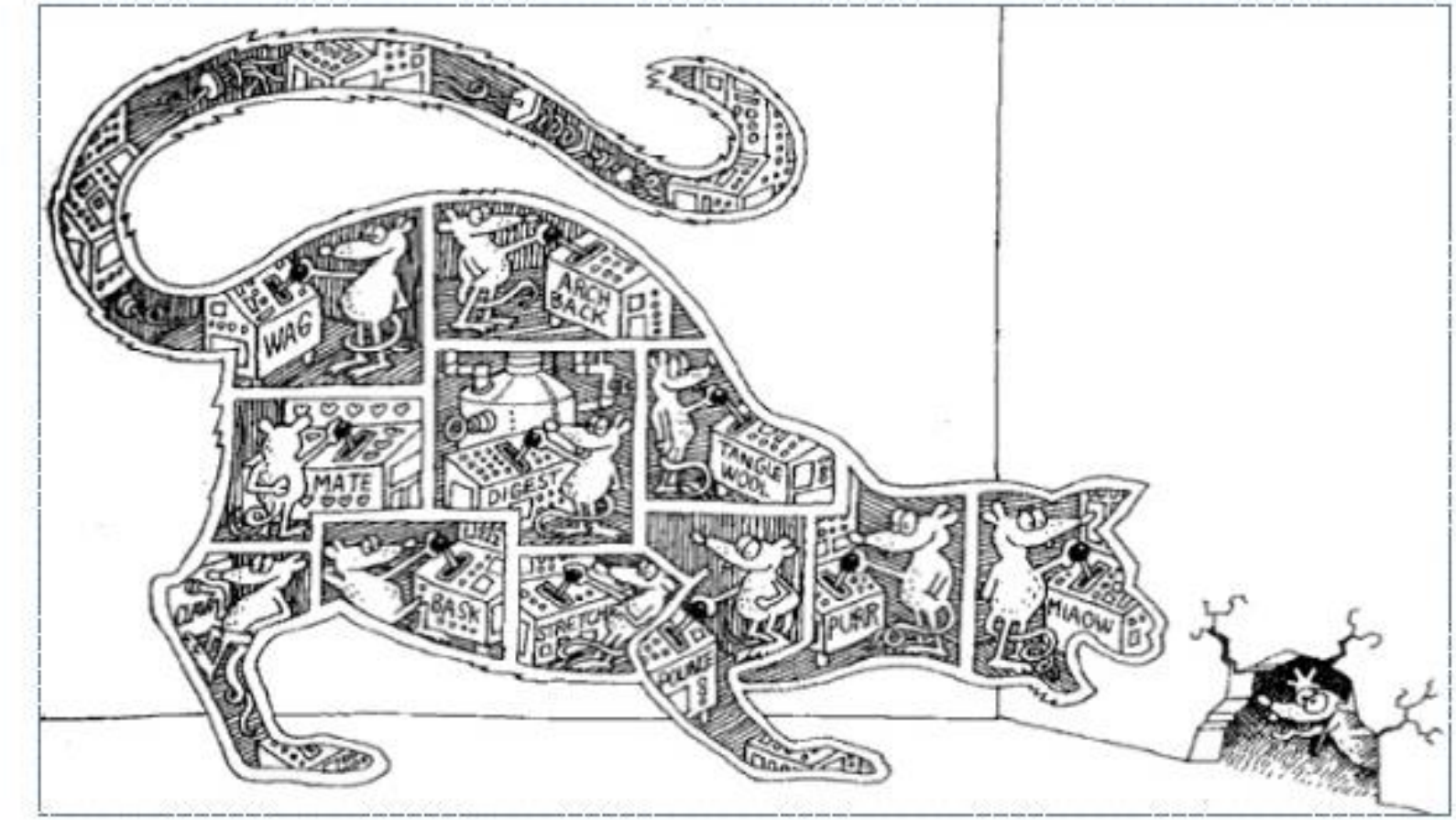
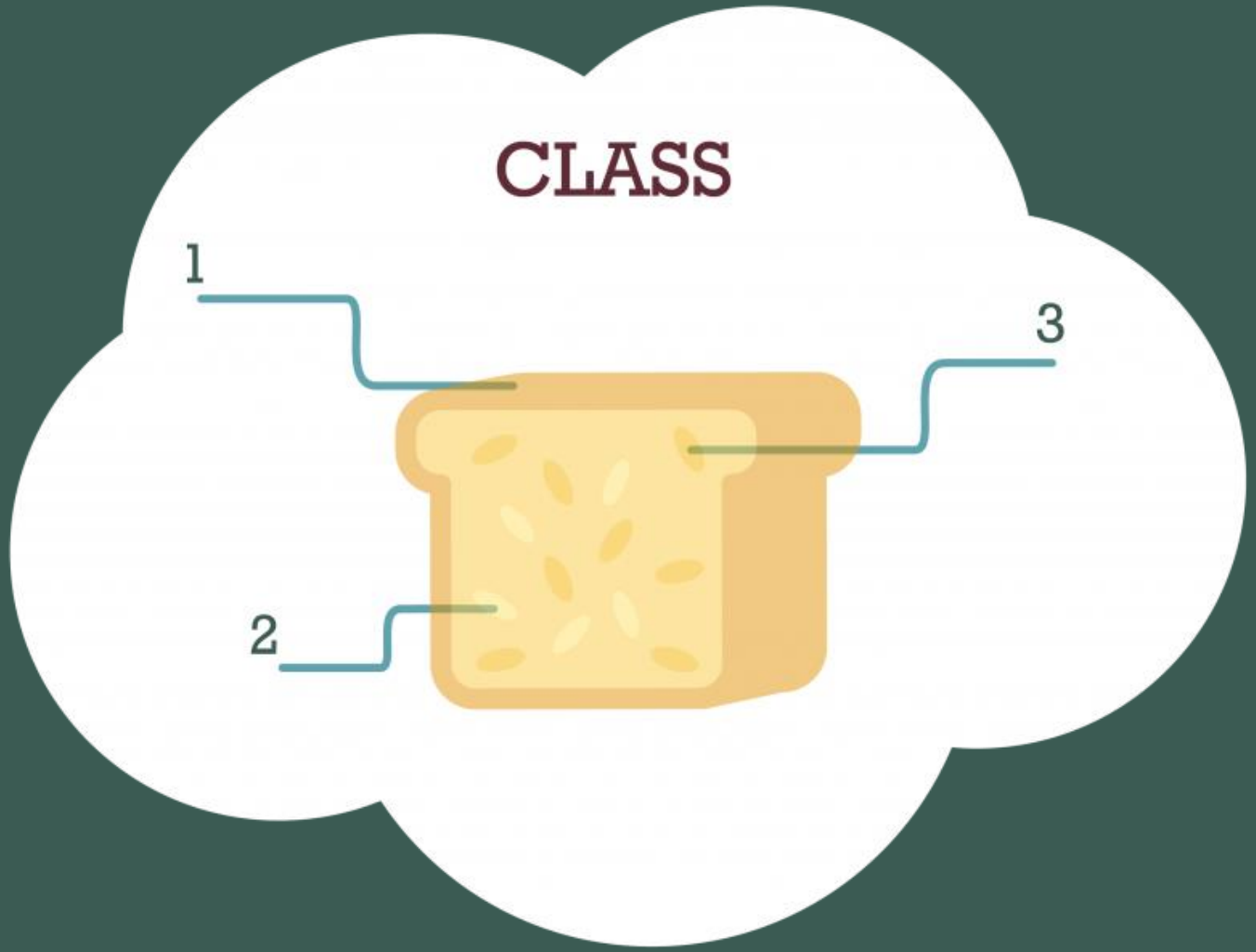
Object Terminology

Why are objects important?



Objects:

- Group details together: data and methods that work on the data
- It is a programmer-made type.



Class File

- The template for all of the variables that will be made using that type.
- It’s name is a type, the file name and the constructor’s name.

Objects have a lot of terminology

Instance Variables

Constructor

Mutator

Accessor

Facilitator

```
public class Item {
    private double price;
    private String name;
```

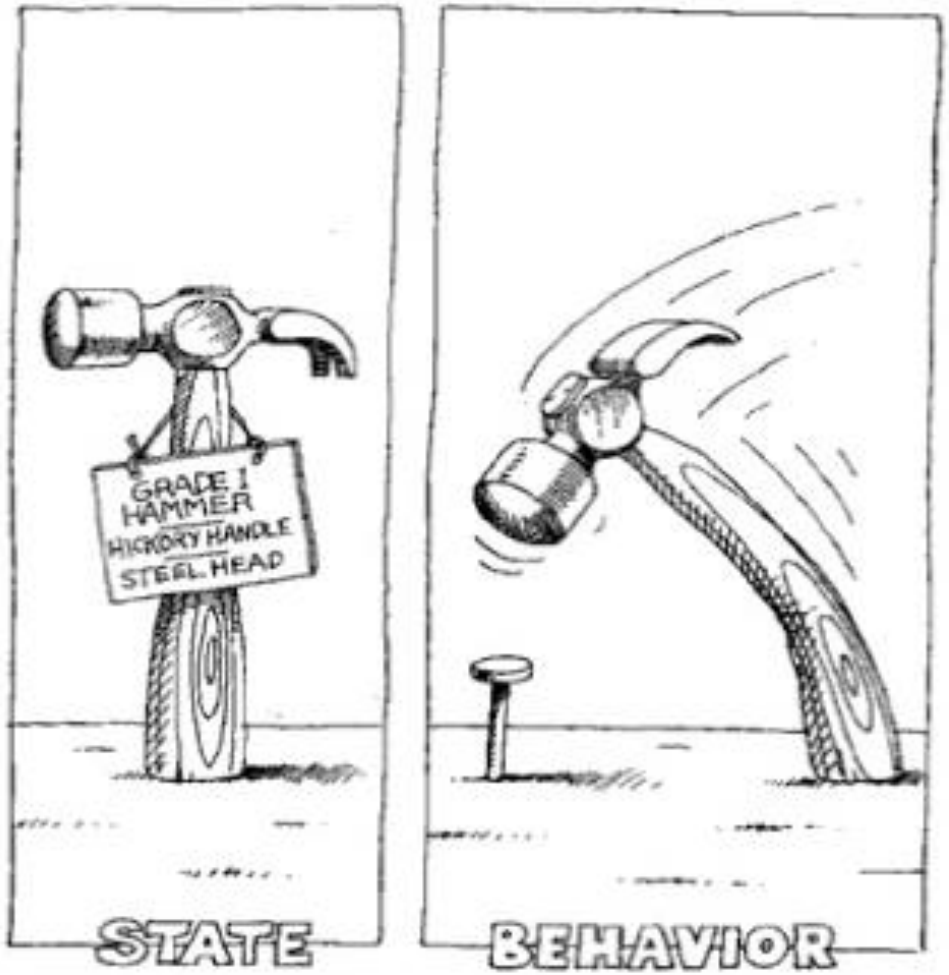
```
public Item(){
  price = 13.45;
  name = "t-shirt";
}
public Item(double p, String n){
  price = p;
  name = n;
}
```

```
public double getPrice(){
  return price;
}
public String getName(){
  return name;
}
public String toString(){
  return "The "+name+" costs $"+price;
}
```

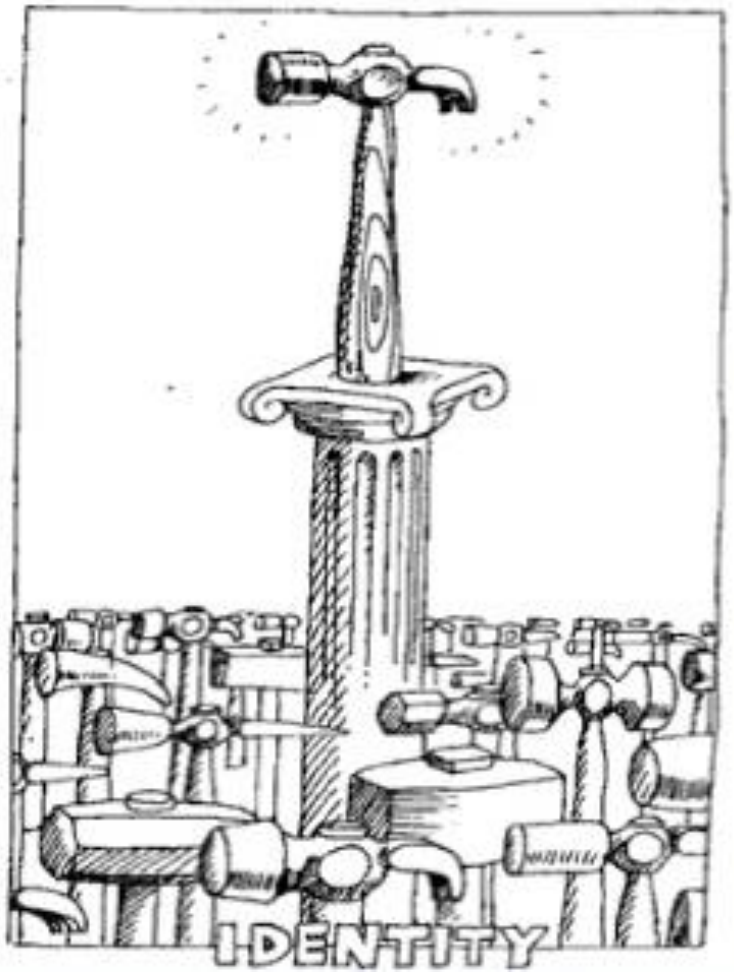
```
public void setPrice(double p){
  price = p;
}
public void setName (String n){
  name = n;
}
```

```
public boolean equals(Item i){
  if(i.getName().equals(name)
          && i.getPrice()==price)
     return true;
  else
     return false;
}
public int compareTo(Item i){
  //on the basis of price
  if(i.getPrice()>price)
      return -1;
  else if (i.getPrice()==price)
      return 0;
  else
      return 1;
}}
```

The Class lays out the data and the methods.



A variable uses the class as it’s type.



```
public void show(View view) {
    TextView textArea = (TextView) findViewById(R.id.TextArea);
    Item shoe = new Item(23.45, "flip-flops");
    Item shirt = new Item();
    textArea.append(""+shoe.toString ());
    textArea.append("\n"+shirt.toString ());
    textArea.append("\n"+shoe.getPrice ());
    shirt.setPrice (100.98);
    textArea.append("\n"+shirt.toString ());
    textArea.append("\n" + shirt.compareTo(shoe));
    textArea.append("\n" + shoe.equals(shirt));
}
```

```
public class Item {
    double price;
    String name;
```

```
public Item(){
  price = 13.45;
  name = "t-shirt";
}
public Item(double p, String n){
  price = p;
  name = n;
}
```

```
public double getPrice(){
  return price;
}
public String getName(){
  return name;
}
public String toString(){
  return "The "+name+" costs $"+price;
}
```

```
public void setPrice(double p){
  price = p;
}
public void setName (String n){
  name = n;
}
```

```
public boolean equals(Item i){
  if(i.getName().equals(name)
        && i.getPrice()==price)
     return true;
  else
     return false;
}
public int compareTo(Item i){
  //on the basis of price
  if(i.getPrice()>price)
      return -1;
  else if (i.getPrice()==price)
      return 0;
  else
      return 1;
}}
```

But, why do we use objects?

They seem like pointless busy work.

Software Development Engineer

SAVE

Amazon
Toronto, ON

BASIC QUALIFICATIONS

- Bachelor's Degree in Computer Science or related field
- 3+ years of software development experience with Java, C/C++
- Experience with Object-Oriented Programming (OOP) and/or Design (OOD)
- Computer Science fundamentals in data structures, algorithms, problem solving and complexity analysis

Generalist Programmer

SAVE

Ubisoft
Toronto, ON

Apply on ZipRecruiter | Apply on Hitmarker | Apply on Jobs | Apply on Jora | Apply on Jobilize LLC | Ap

Over 1 month ago | Full–time

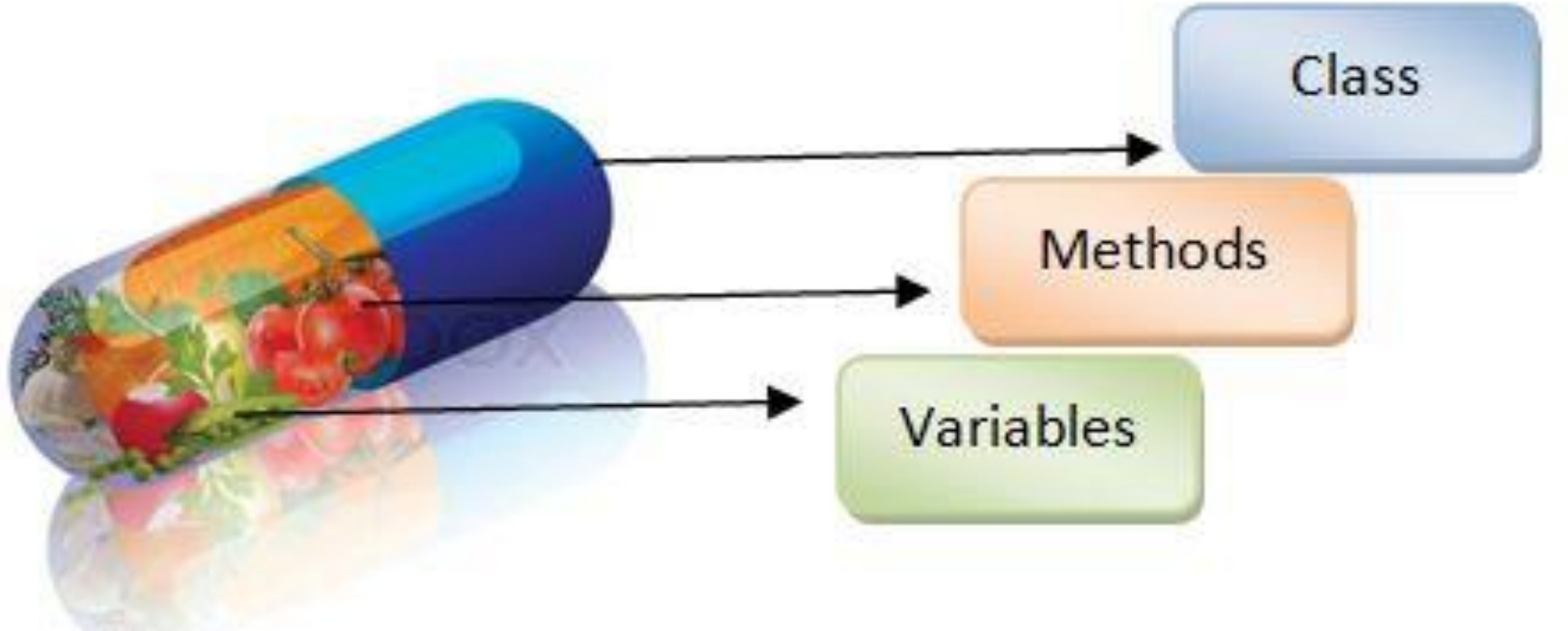
What it takes to make it
• Bachelors in computer science or computer engineering, or equivalent training or experience;
• Minimum 3+ years' software programming experience, in the video game industry;
• Knowledge of the workings of video game engines;
• Interest and experience in most aspects of game technology development (audio, 3D, animation, gameplay, UI, network and tools);
• Experience using one or more commercial game engines;
• Experience of refactoring and optimization.
• Solid proficiency in C/C++ language and debugging;
• Ability to collaborate and work well with other disciplines;
• Understanding of performance sensitive programming (algorithmic and low level optimizations);
• Good understanding of Software Engineering principles, e.g. Object Oriented programming, etc.

Encapsulation

1. An object's code is self-contained and independent of other code.
2. It relies only on itself.
3. Objects are easy to move around and use by other coders.
4. Objects are easy to update.

Encapsulation is essentially organization.

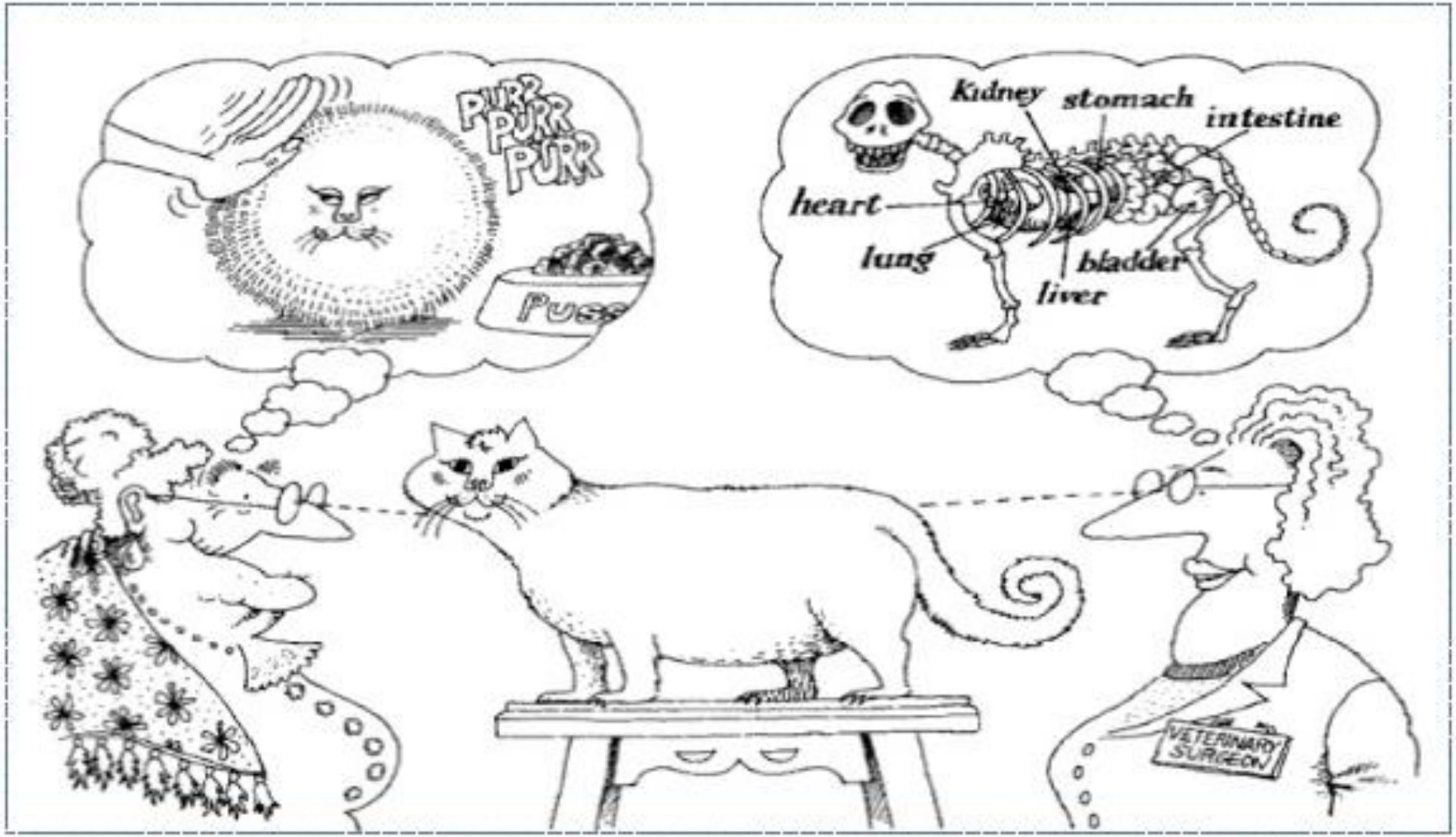
The independence of other code means that work can be divided in a large team.



Abstraction:

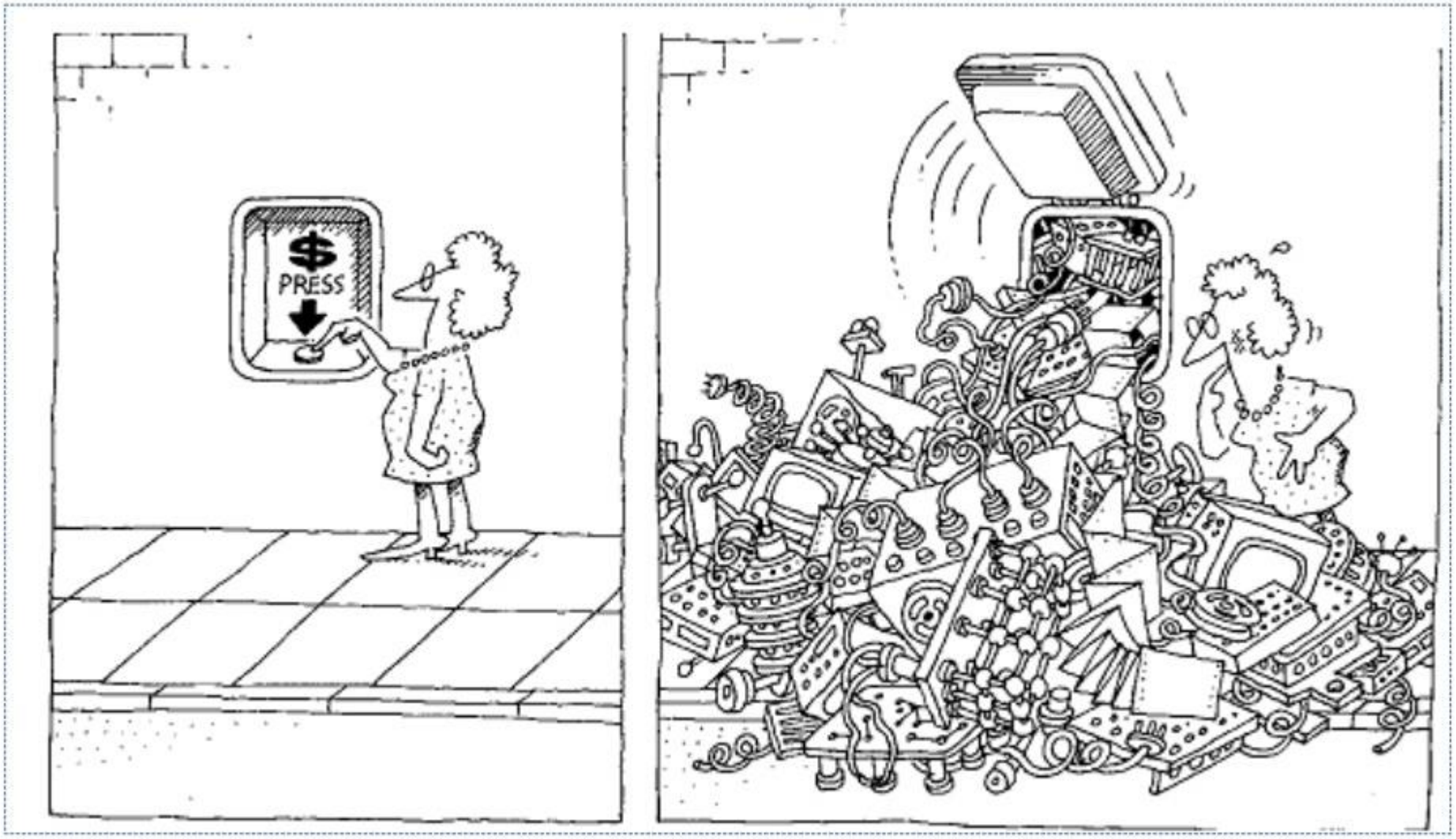
- Makes it easy for other programmers to instantiate in their own programs.
- Allows other programmers to think of the problem at a higher, more removed level.

Abstraction

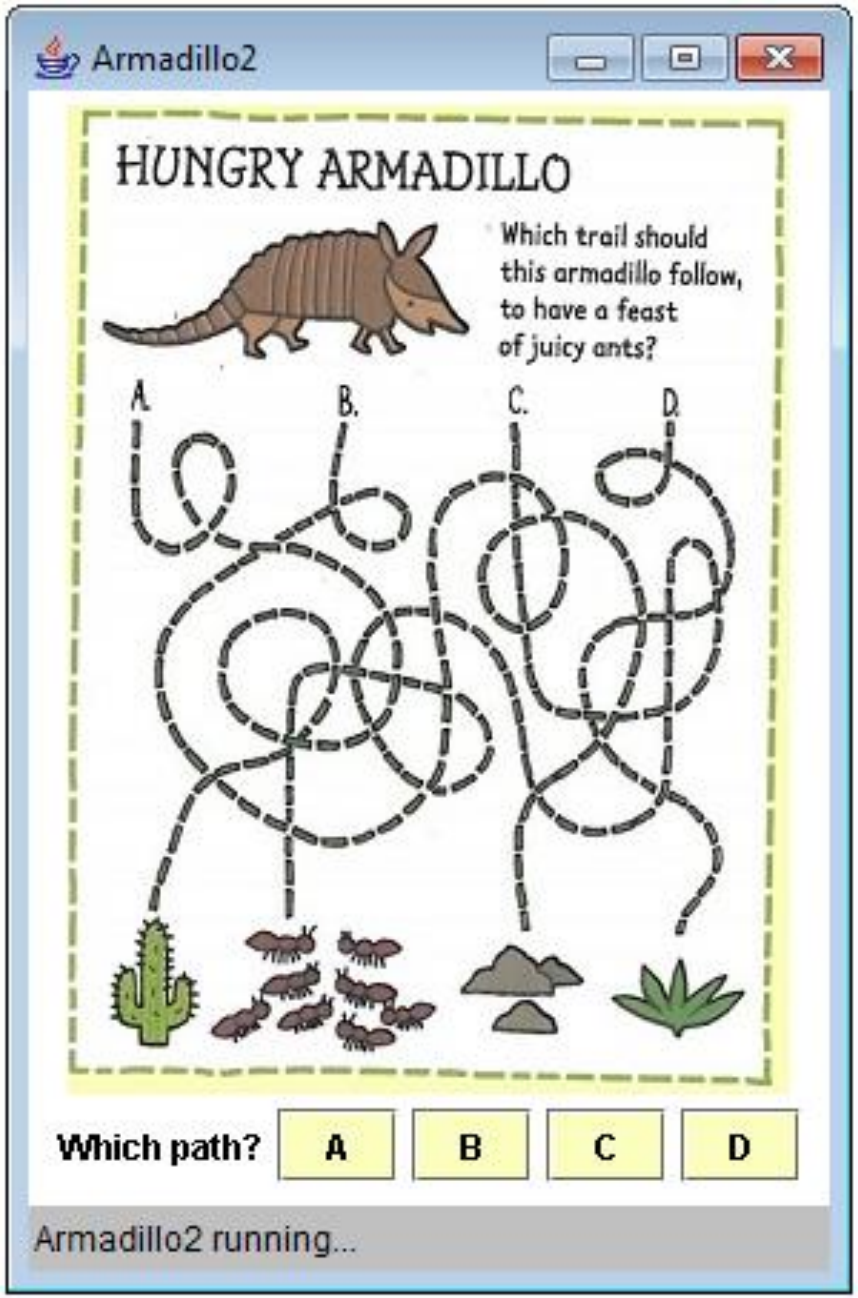


Coder who uses the class

Coder who makes the class



With abstraction, using the class is easy because you don’t need to understand the details of how the class is implemented.



Remember last year? In the applets unit, you could use a JButton class with no knowledge of how to code it.

That's Abstraction!

That was because JButton was an object coded in a class.

Information Hiding – removing details of the instance variables.
The IV are private! They can only be used through accessors and mutators.

You can’t get at the variables in a class directly. It keeps the code more stable. Other coders, who might not understand your brilliant system, can’t easily mess with the variables. They are only allowed in via mutators and accessors.