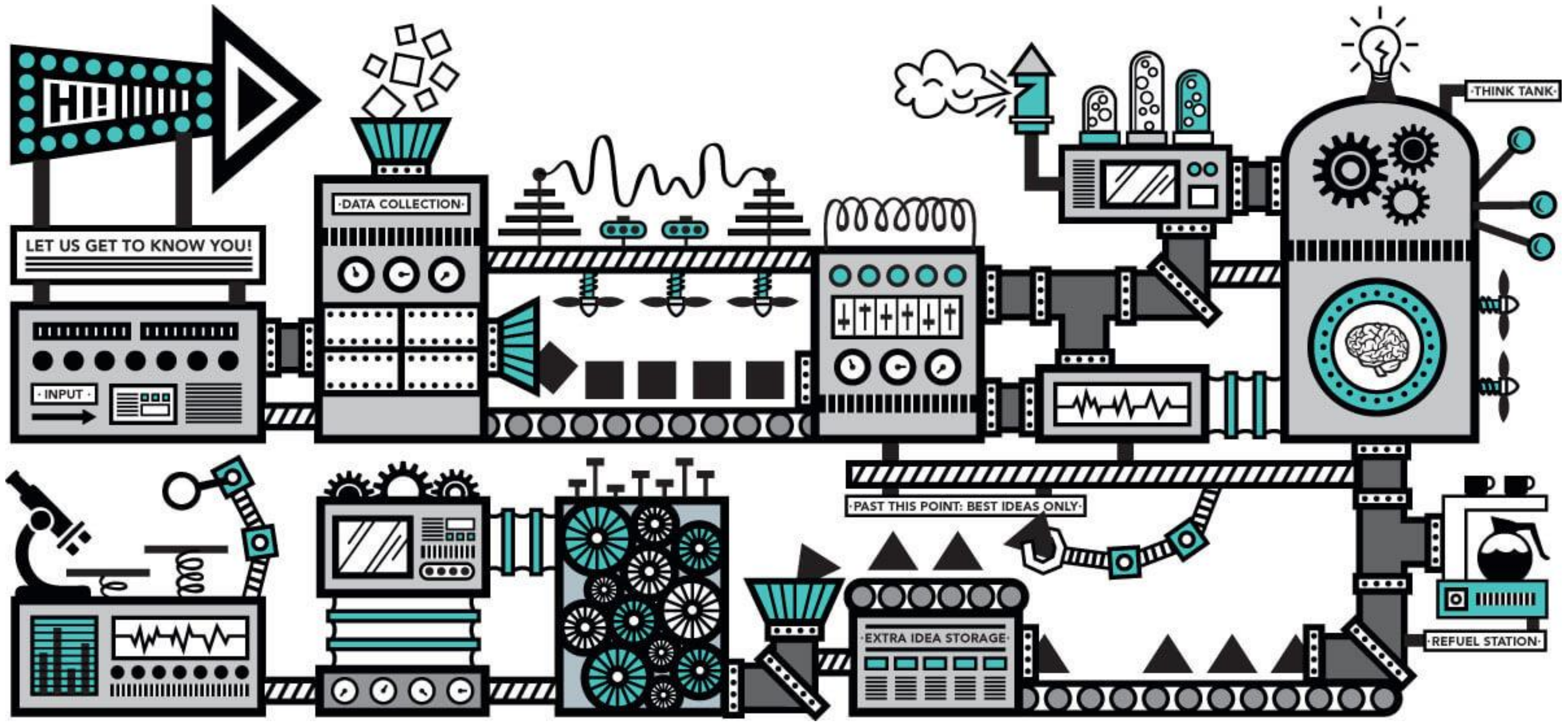
Defining Methods Part II

Sect. 3.3, 8.2

There's a method in my madness.

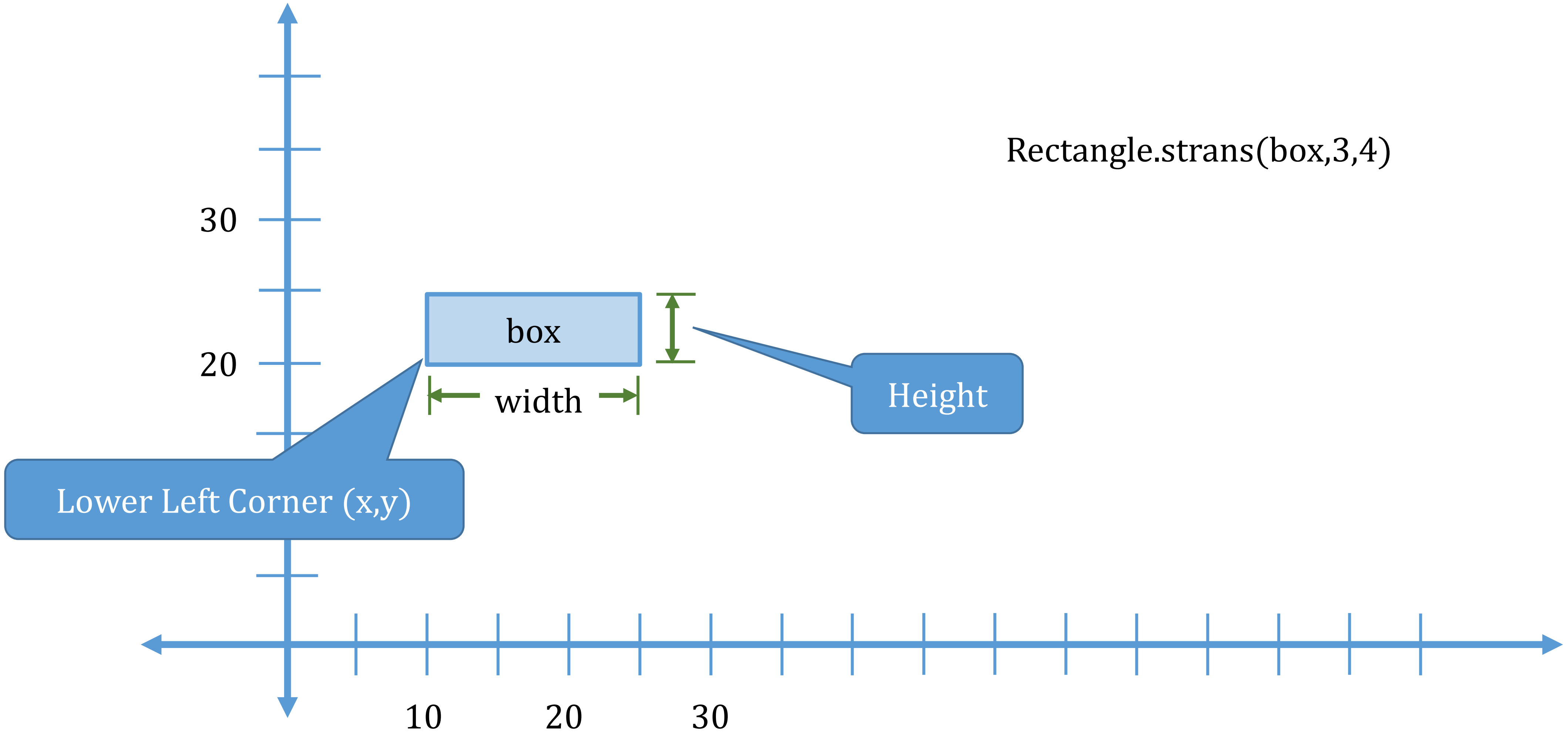


static methods behave like C

```
class XmpStatic {
    static int add3(int x) { return x+3; }
}
```

- Use input parameters (x) to determine returned result
- Not explicitly object oriented
  - No implicit references to fields in the class!
  - May have reference variable parameters

Modeling a Rectangle



Rectangle.strans(box,3,4)

Instance vs. Class Methods

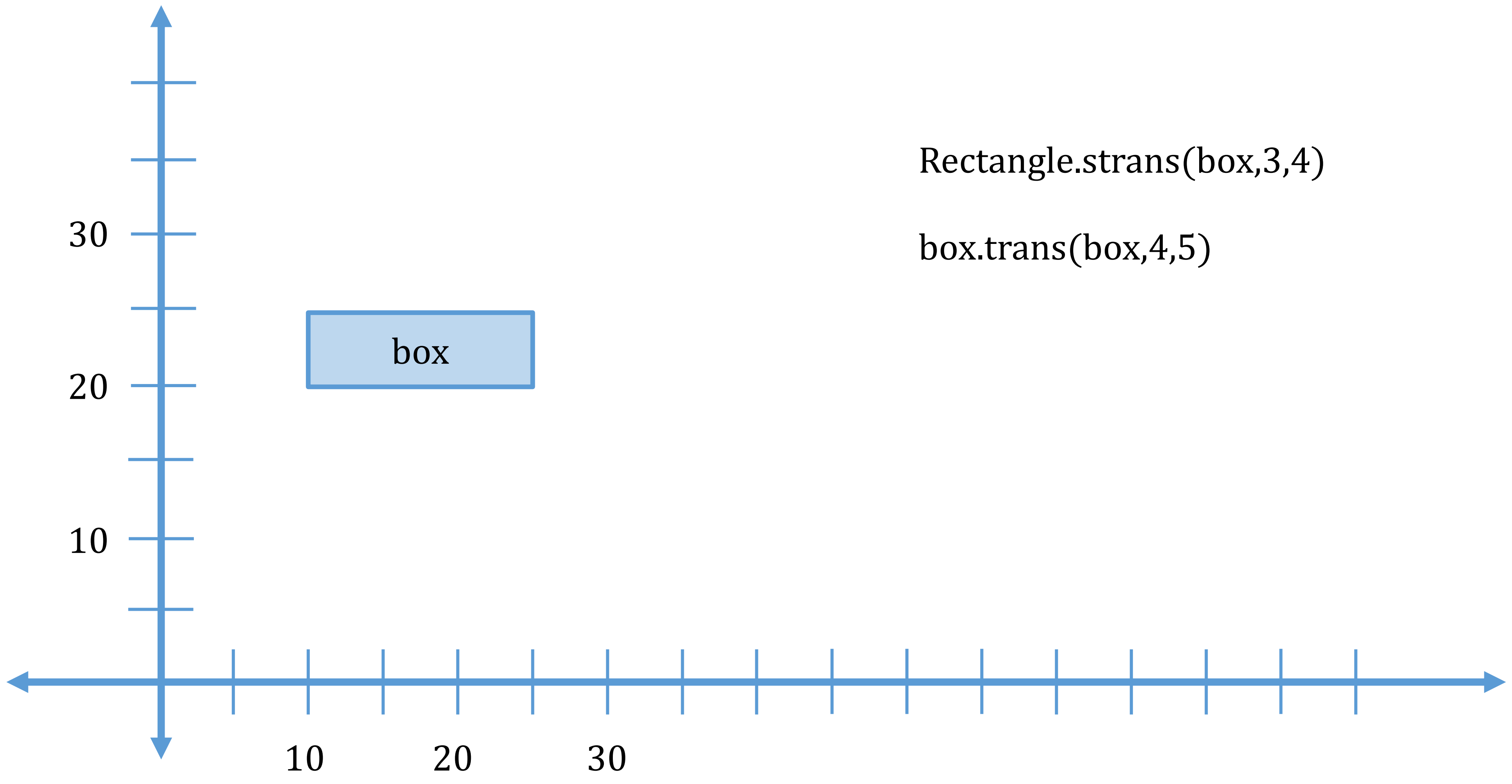
- If a method is declared as `static`, it is a **class method**
  - Invoked like a function… not as an action on an object
  - Class methods do not have a receiver object, and therefore do not have a `this` implicit parameter
- Most methods are **instance methods** – methods which are invoked as actions "on" specific receiver objects
  - Do not have a `static` modifier!
  - Have an implicit `this` reference parameter to reference the receiver object
  - Can implicitly reference fields in the `this` object.

Instance Method Implicit this Parameter

- The object that "receives" the action (the object which the action is performed on) is called the "receiver" object.
- The receiver object is ***implicitly*** placed in the parameter list of the method, as if (but not actually) you had specified:

```
class Rectangle {
        static void strans(Rectangle rect, int dx, int dy) { }
        // is very similar to…
        void trans(int dx, int dy) { … }
}
```

Modeling a Rectangle



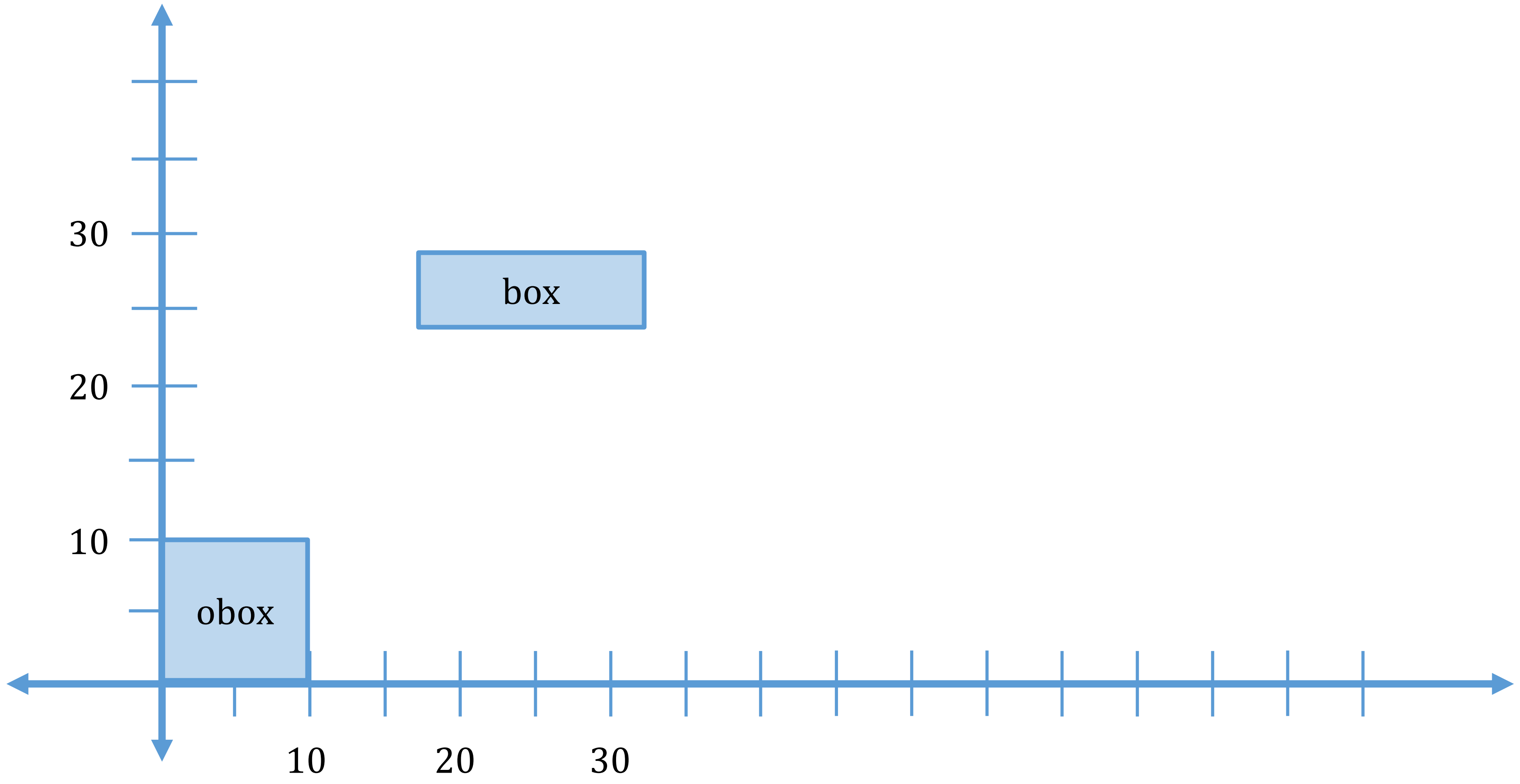
```
Rectangle.strans(box,3,4)

box.trans(box,4,5)
```

Method “Signatures”

- Java allows multiple definitions of the same method name!
  - As long as the parameters types are different!
- Method signature includes: name **and** parameter types
- When a method is invoked, Java looks at the arguments to determine the signature, and invokes the method with that signature!
- Often multiple constructors with different signatures
- Sometimes used for instance or class (static) methods as well

Modeling a Rectangle



Accessor vs. Mutator Methods

Sect. 2.5

- If a method changes the receiver object field values, it is called a ***mutator*** method.
  - For example… the "trans" method in Rectangle changes the value of x and y… it is a mutator method
- If a method does not change the receiver object field values, it is called an ***accessor*** method.
  - For example… a "getWidth" method in Rectangle does not change any of the fields in rectangle… it is an accessor method.
- If all methods are accessor methods, the Class is called an ***immutable*** class
  - For example… the String class is immutable.

Demonstrating Immutable Strings

```
String str = “This is a test.”;
String str2 = str.replace(‘t’,’v’);
System.out.println(str);
System.out.println(str2);
```

```
This is a test.
This is a vesv.
```