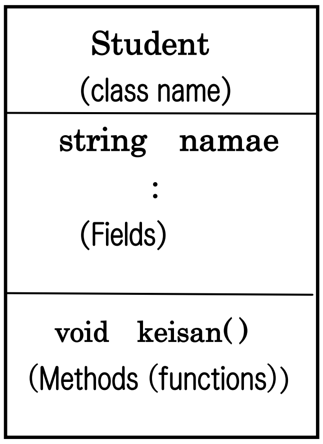
Episode 14 (Key Points of the Java Language)

Tanuki, to be able to use the Java language, it is absolutely necessary to understand the terms class, instance, and constructor and what they mean. Otherwise, you'll just be a worker attaching parts together without being able to see the big picture.

Fox, I understand that classes, instances, and constructors are important, but what is a class?

A class is like a C language structure. However, a structure only contains member variables (data), while a class is a package of member variables plus processing (functions). To make classes easier to understand, we discussed the C language in Episode 13. A class is like a structure, it is only a framework. It is often likened to a TAIYAKI mold. The mold is the class. From this mold, you can make a TAIYAKI with bean paste, cream, or jam. The resulting variety of TAIYAKI is called an instance (entity) or object (thing) because it is an actual TIYAKI that can be eaten. Since the mold of a TAIYAKI cannot be eaten, it is called a class. Creating an object is also called instantiation. The image of a class is shown in the figure on the left.

Class Diagram



The above class diagram can be represented in a Java program as shown on the right.

```
class Student{
    String namae ;
    int kokugo;
    int sugaku;
    int goukei;

  void keisan(String simei,int ka2,int ka3){
     namae = simei;
     kokugo = ka2;
     sugaku = ka3;
     goukei = kokugo + sugaku;
    }
}
```

I see. So where does this Student class exist?

This is just a framework without substance, so it's an area to store the program, or code area in memory.
Next, we have constructors and instances.
The following substitution formula is important.

Student seito 1 = new Student( ) ;

(Class) (reference type variable) (new operator) (constructor)

The function (method) Student( ) on the right is the constructor.

Where is a constructor a function that is created? I don't think I saw any mention of a constructor anywhere in the source file, but you can find it at ・・・・!

Constructors that are not defined by the programmer are automatically created when the source files are compiled, and are called default constructors. Of course, the programmer may define constructors explicitly. Because they are created automatically at compile time, the class name (Student) becomes the constructor name.
The new operator calls the constructor (Student( )) to create an instance (entity). It is the same as a pointer variable, but the address of the instance created by the constructor is stored in the reference type variable (seito1). The constructor can create an instance with no arguments, and then assign data to the member variables later, or it can set arguments from the beginning.

Kitsune, if the constructor is automatically created at compile time, that's good because it probably exists in the executable file, but I think you can create any number of instances, seito1, seito2, seito3, but in what area of memory are they created?

Tanuki, you've got a good point. Instances are created in the heap area of memory. The first address of each instance in the heap area is stored in the reference type variables seito1, seito2, seito3. However, the variables and values of the functions (methods) in the instances are placed in the stack area. Do you get it, raccoon dog? Let me show you the following diagram so that you can understand it a little better.

Program Example（Let it be Rei.java.）

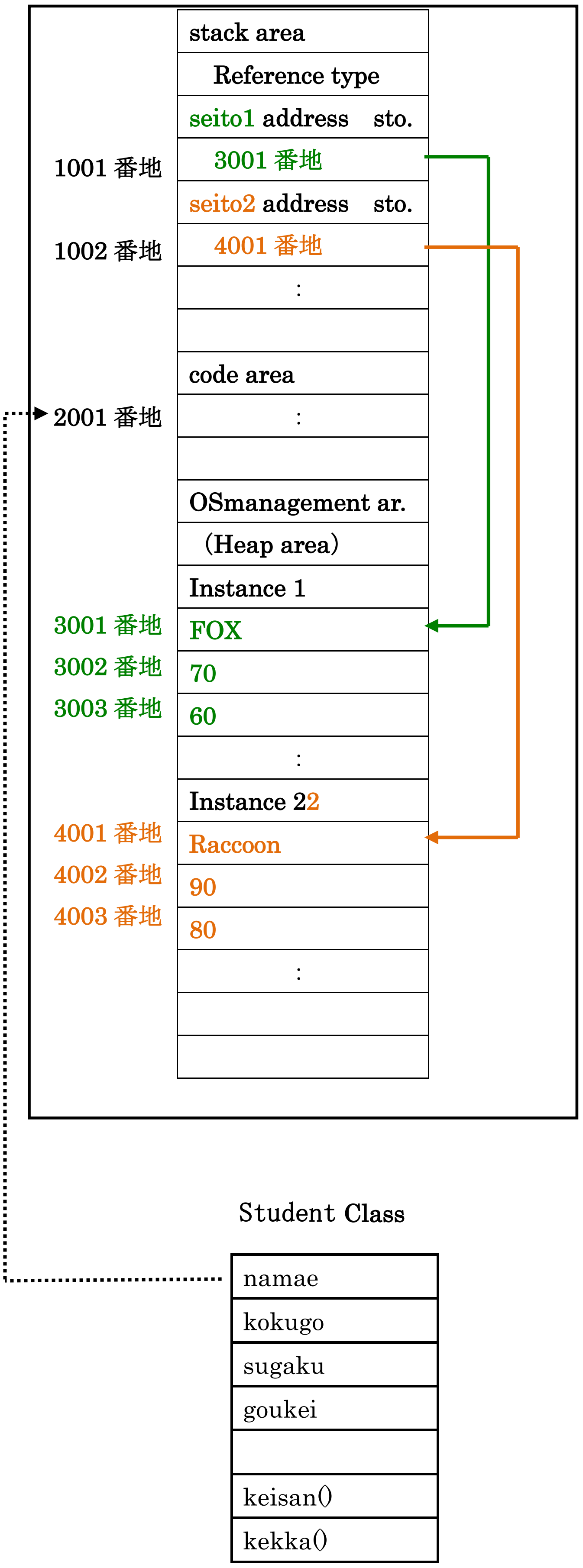
```
public class Rei{
   public static void main(String args[]){
      Student seito1   = new Student();
      seito1.keisan("FOX",70,60);
      seito1.kekka();
      Student seito2   = new Student();
      Seito2.keisan("Raccoon",90,80);
      Seito2.kekka();
    }
}

class Student{
     String namae ;
     int kokugo;
     int sugaku;
     int goukei;

   void keisan(String simei,int ka2,int ka3){
      namae = simei;
      kokugo = ka2;
      sugaku = ka3;
      goukei = kokugo + sugaku;
    }

  void kekka(){
   System.out.println("namae:" + namae);
   System.out.println("goukei:"+goukei);
   }
}
```

Memory storage image



The Java language is not easy to understand if you only look at the program (Rai.java). It may be easier to understand if you think of it in combination with the image of memory storage on the right. In the second line of the program, there is the main( ) function, which is the first function executed in the program, just like in C. In the third line, the entity (instance) of OIRA (seito1) is created. It is the reference type variable seito1 that tells you that the entity of OIRA is at [address 3001]. A reference type variable is the same as a pointer variable in C language. Note that seito1 is created in the stack area, and the entity (instance) of OIRA is created in the heap area.
The heap area belongs to the OS management area.
The constructor (Student( )) invoked by the new operator creates the lla entity while looking at the Student class defined after line 12.
Similarly, the raccoon dog entity is created on line 6 and its location is stored in seito2.
The Student class in the lower right is only a layout, so it is stored in the code area that contains the program.

Kitsune, you illustrated the memory area, so I think I understand it somewhat. But I'm not sure about the stack area or heap area!

The stack area has a fixed amount of memory to be allocated. It is mainly used to store the return address from the main program to the subprogram when control is transferred from the main program to the subprogram. You will study the stack area in the information exam, along with the terms PUSH (take out) and POP (store).
The heap area is an area of memory that the programmer can explicitly allocate as needed and release when finished using it.

I see! The more you study, the more you know what you need to study next? So, the more I study, the more things I don't know, and the more I feel humbled that I don't know anything.
When I studied PUSH and POP, I was taught the "first-in, first-out" method for inserting and outputting data, but I did not understand the connection with programs. I was able to answer the questions on the information exam, though.

Tanuki, you have grown up!
Furthermore, the stack area has a fixed and small amount of memory allocated, so if you keep going into subprograms without retrieving the stored return address, you'll get an error message "OUT OF MEMORY" (memory area is flat) and the program will freeze. At such a time, you may wonder why the program freezes when the program you are running is small and has 2GB of memory.
Maybe I'm getting a little off topic.
I'm persistent, but I'll repeat it again.
The point of the Java language is, understanding classes, instances, and constructors.
Let's talk about the features of the Java language in

Episode 15.

Translated at DeepL