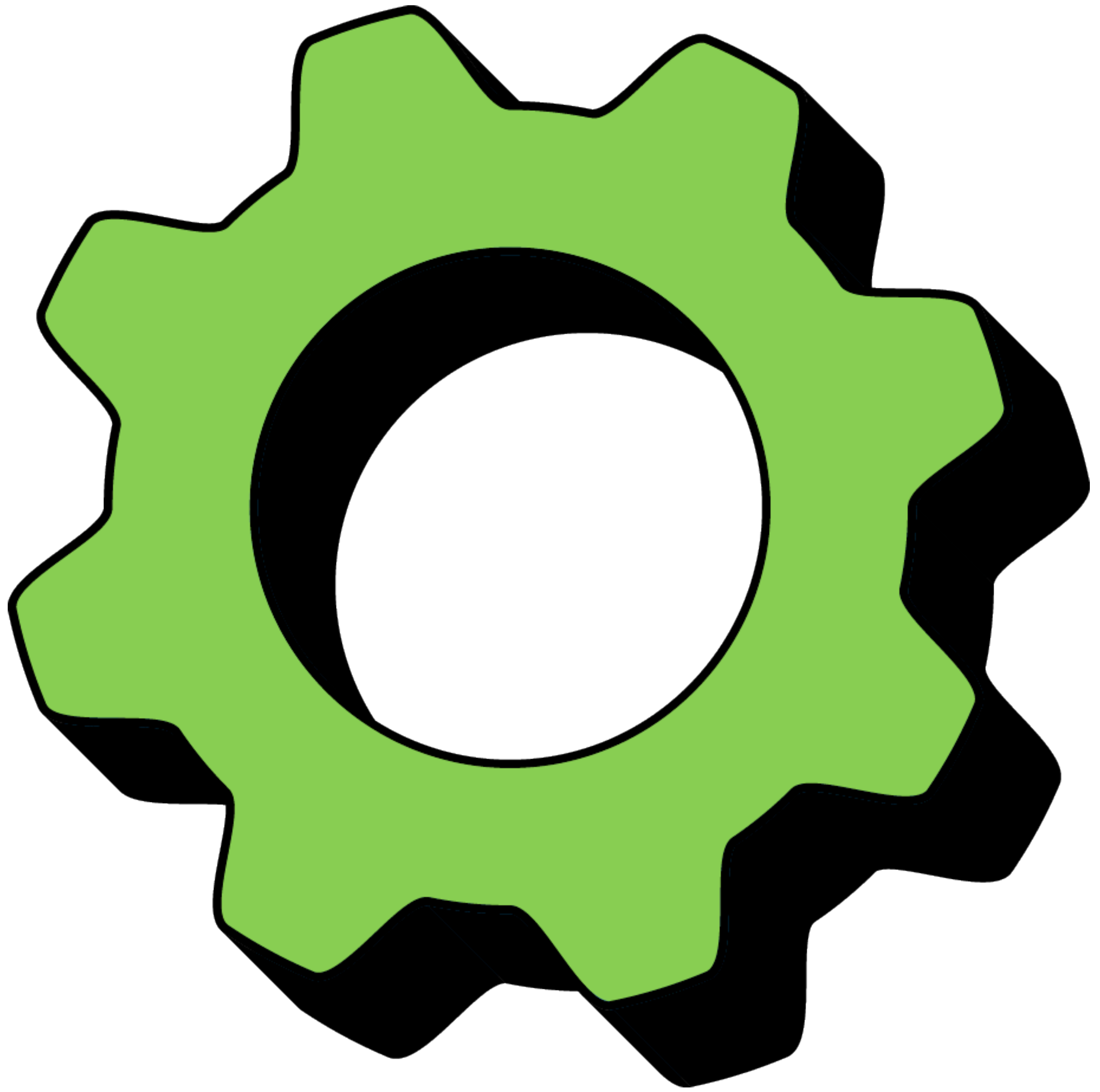
Essential Computing 1

Value methods



Clarification

a method call sends **arguments**
a method has **parameters**

```
public static void main( String[] args ){
    make( 5 );
}
```

argument

parameter

```
static void make( int a ){
    // Missing implementation.
}
```

Terminology "invoking a method" same as "calling a method".

method invokation →

```
public static void main( String[] args ){
    make( 5 );
}
```

method definition →

```
static void make( int a ){
    // Missing implementation.
}
```

Value method: A method that **returns** a value

```
static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

The **return type** defines what type of value the method will return.

```
static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

The **return statement** will exit the method and return the value (here, the result of an expression).

```
static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

The type of the value returned must match the **return type**.

```
static double test(){
    return "lars";
}
```

Compile Error!

The type of the value returned must match the **return type**.

```
static String test(){
    return "lars";
}
```

The type of the value returned must match the **return type**.

```
static double test(){
    return;
}
```

Compile Error!

The type of the value returned must match the **return type**.

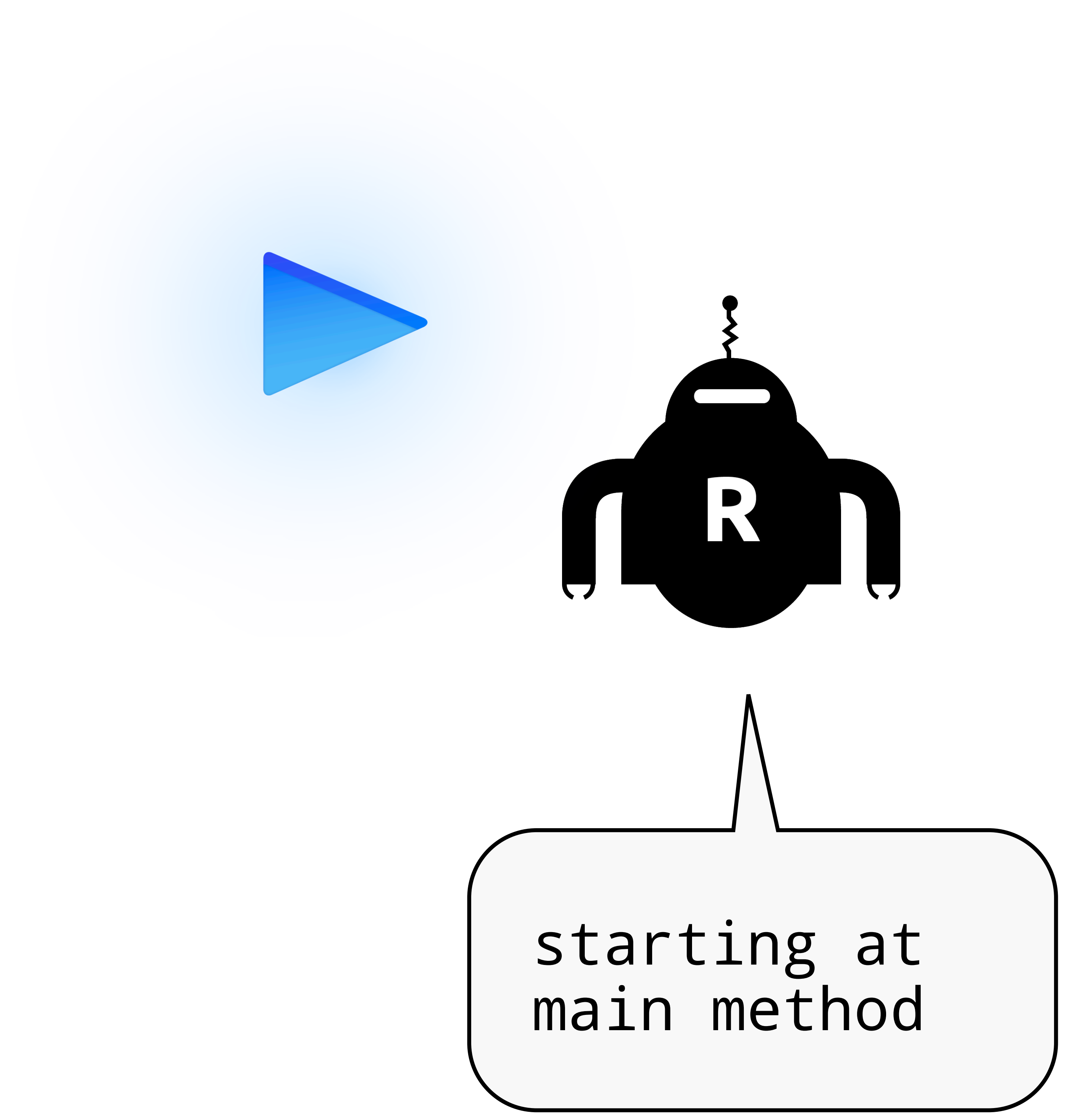
```
static void test(){
    return;
}
```

The return statement is also useful for exiting a method early.

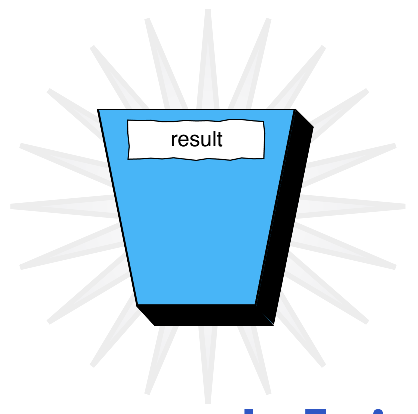
```
static void printOnlyEvenValues( int value ){
    if( value % 2 != 0 ) return;
    System.out.println( value );
}
```

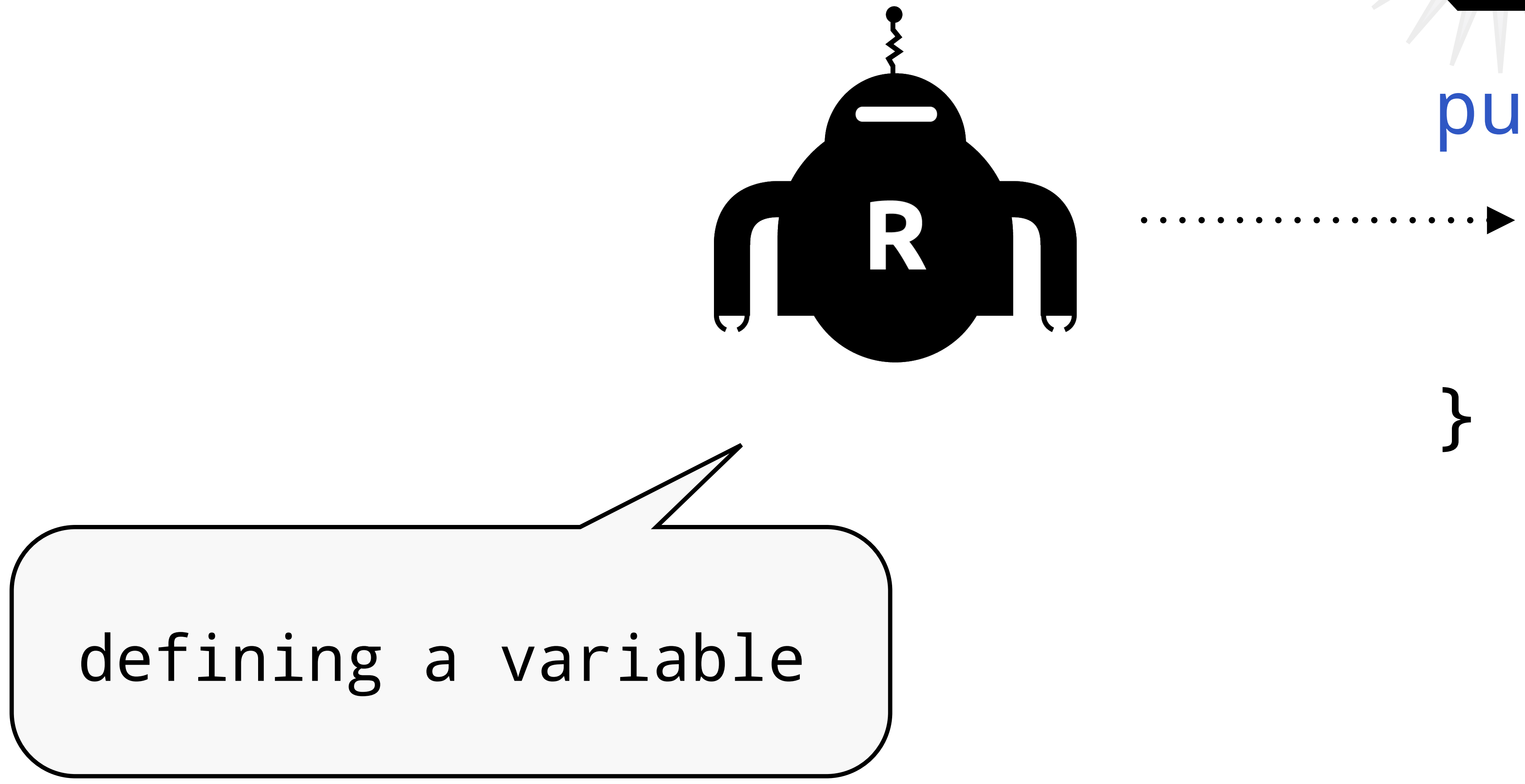
Example

```
public static void main( String[] args ){
   double result = circumference( 1.5 );
   System.out.println( result );
}



static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

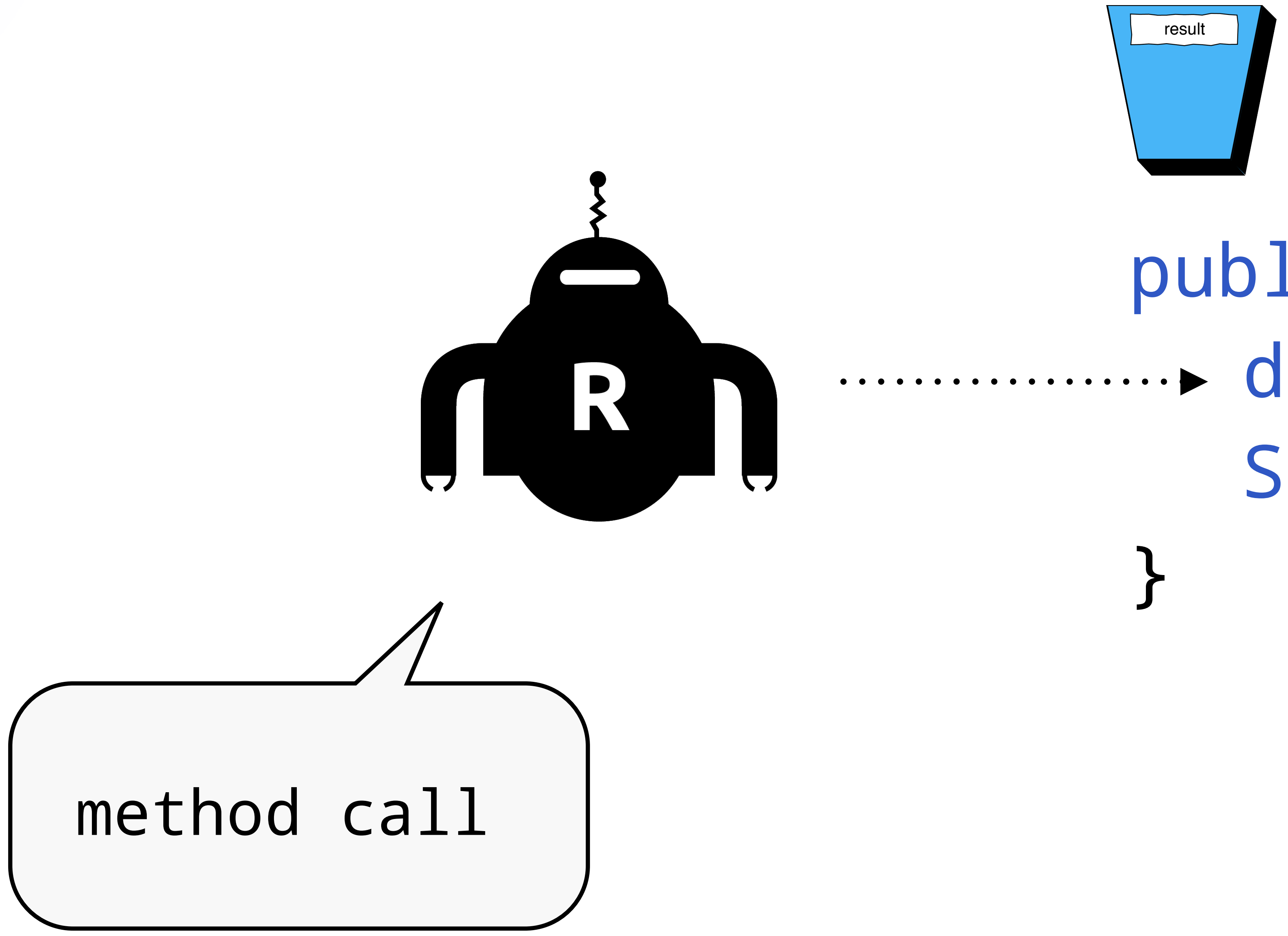


```
public static void main( String[] args ){
   double result = circumference( 1.5 );
   System.out.println( result );
}


static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

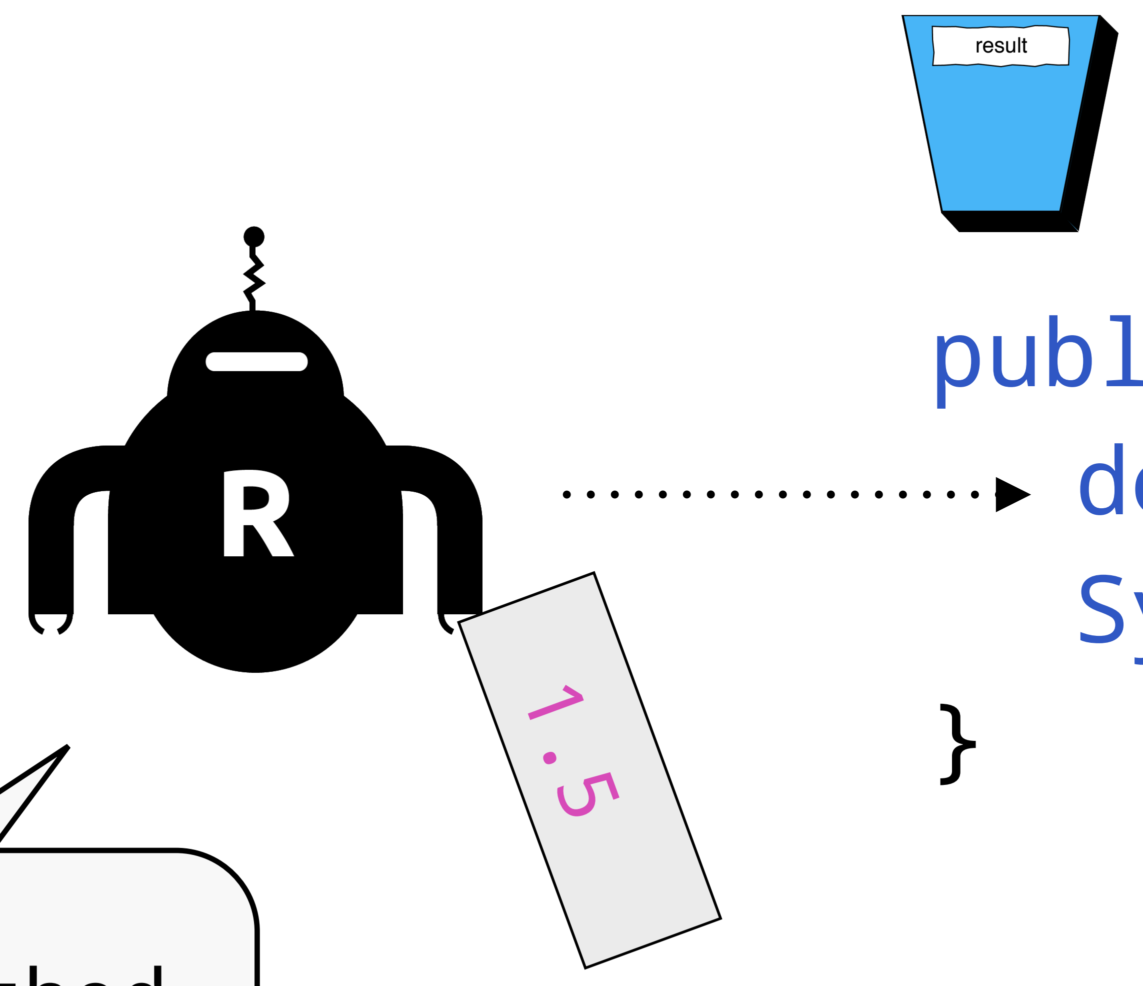




```
public static void main( String[] args ){
  double result = circumference( 1.5 );
  System.out.println( result );
}



static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

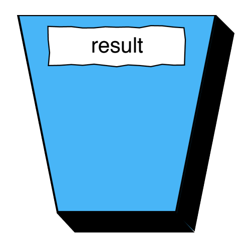


```
public static void main( String[] args ){
  double result = circumference( 1.5 );
  System.out.println( result );
}


static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

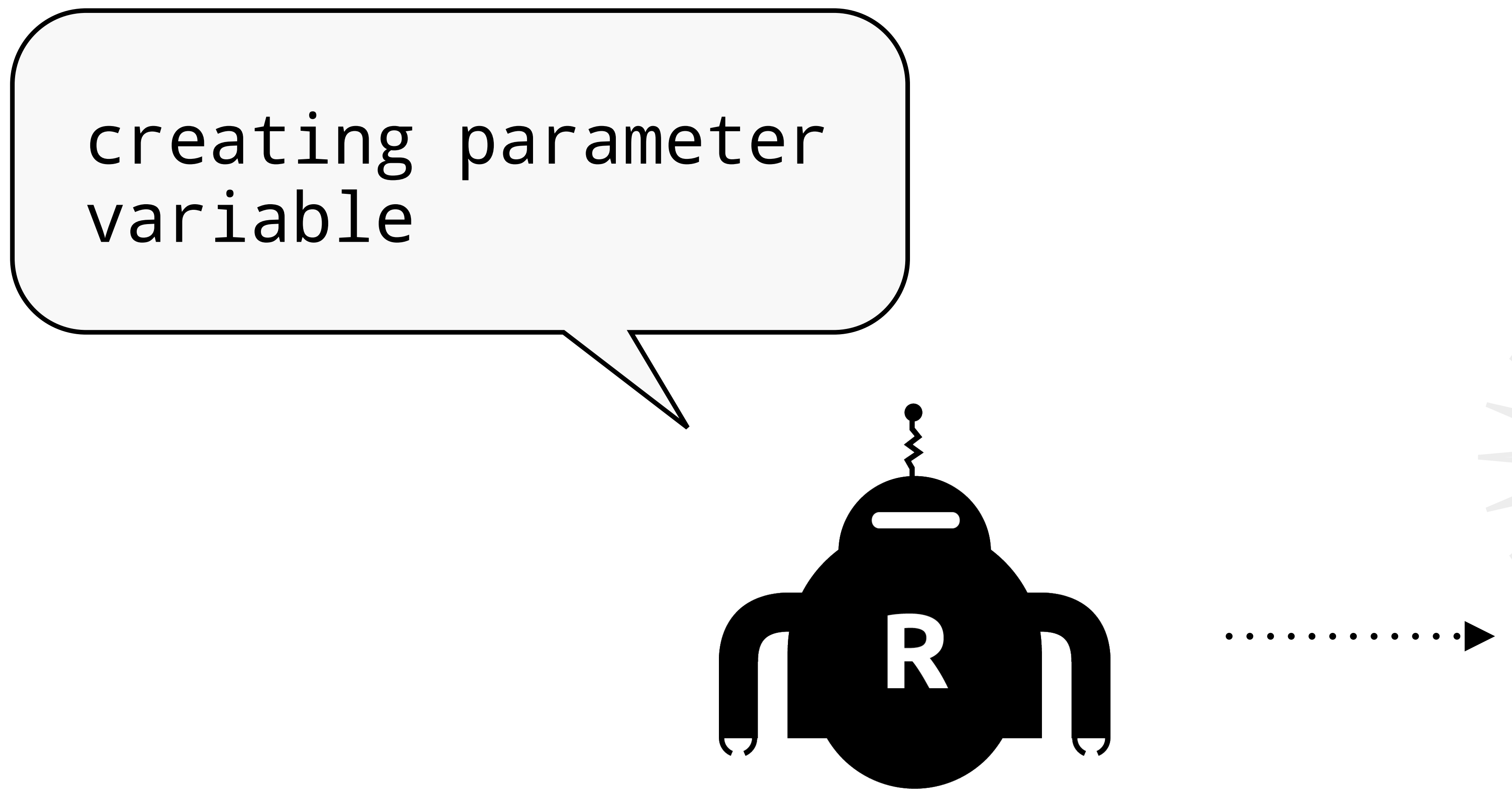


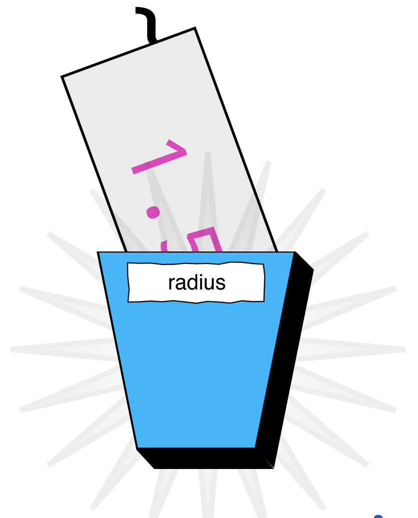
picking up method argument

```
public static void main( String[] args ){
    double result = circumference( 1.5 );
    System.out.println( result );
}


static double circumference( double radius ){
     return radius * 2 * Math.PI;
}
```

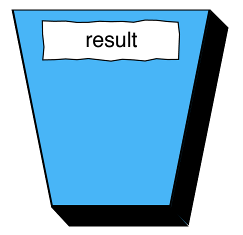


```
public static void main( String[] args ){
 ● double result = circumference( 1.5 );
   System.out.println( result );
}
```

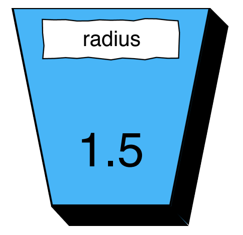




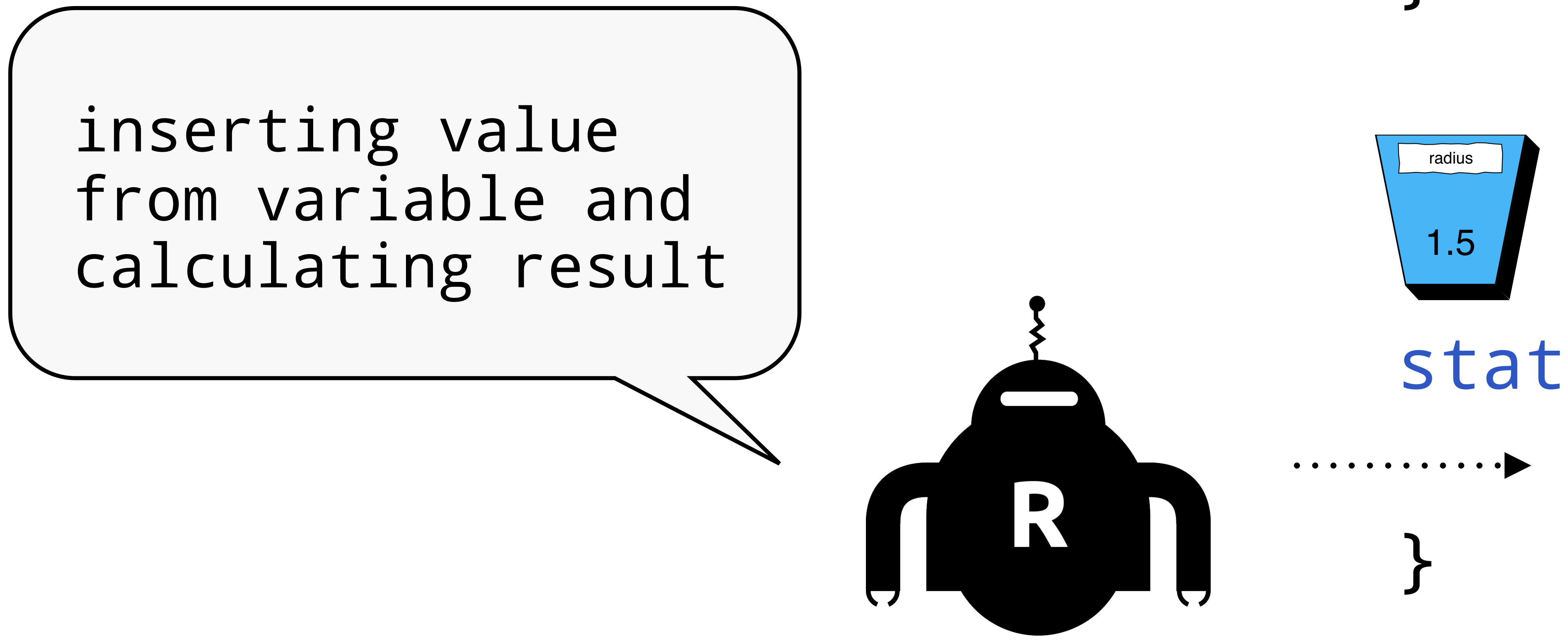
```
static double circumference( double radius ){
     return radius * 2 * Math.PI;
}
```

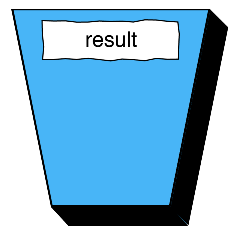


```
public static void main( String[] args ){
 ● double result = circumference( 1.5 );
   System.out.println( result );
}
```

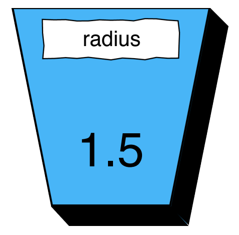


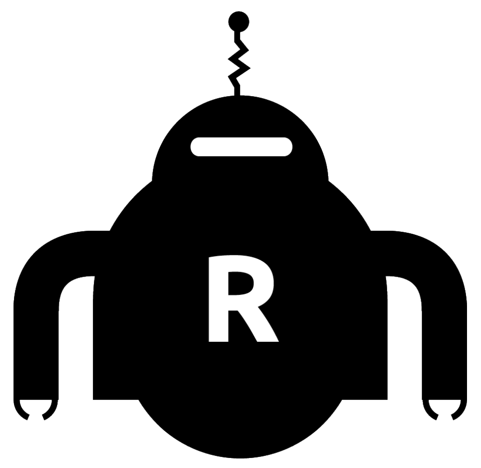
```
static double circumference( double radius ){
   return radius * 2 * Math.PI;
}
```



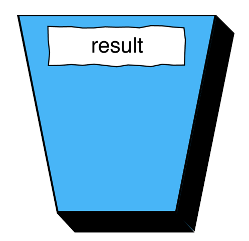


```
public static void main( String[] args ){
 ● double result = circumference( 1.5 );
   System.out.println( result );
}
```

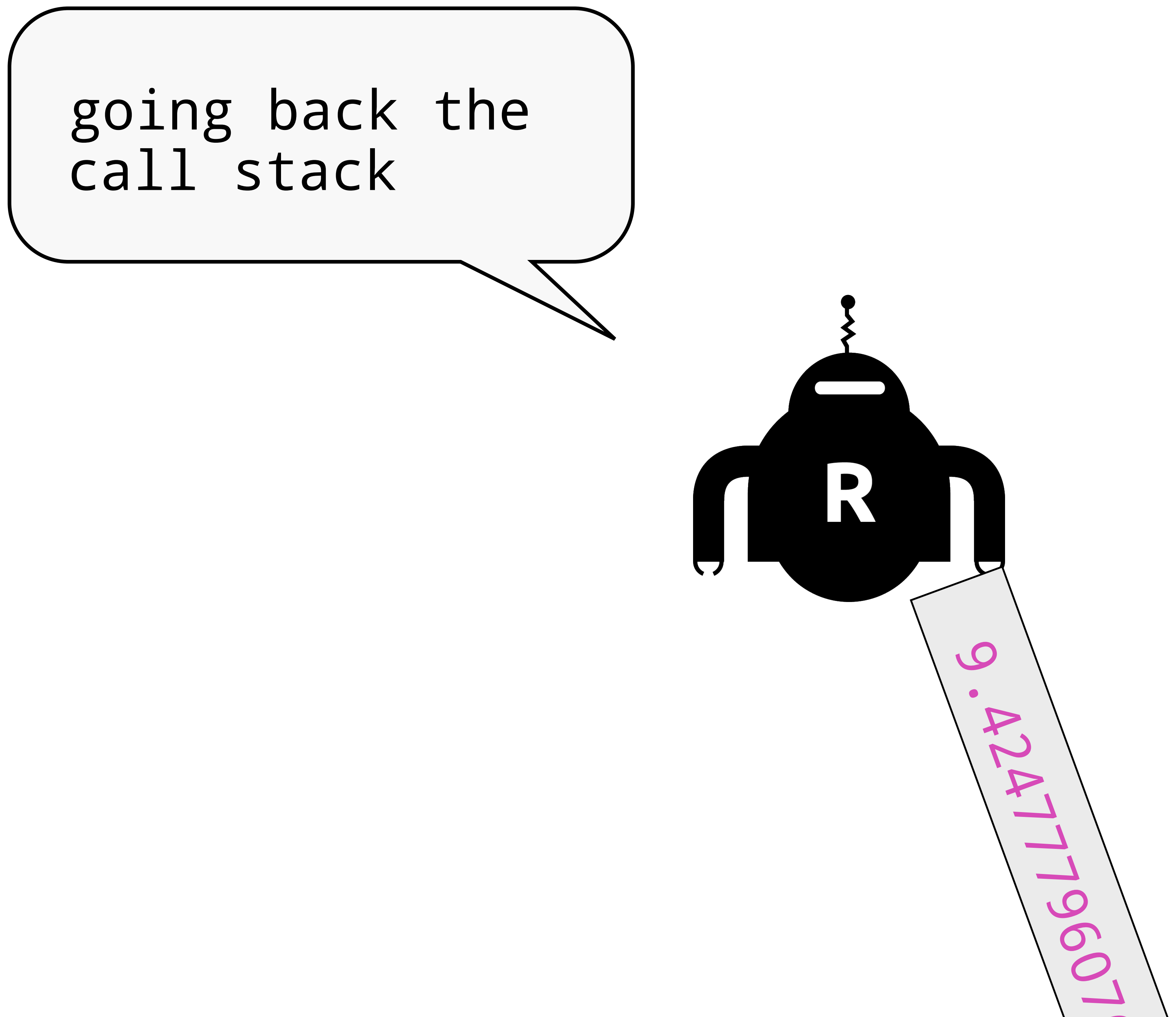




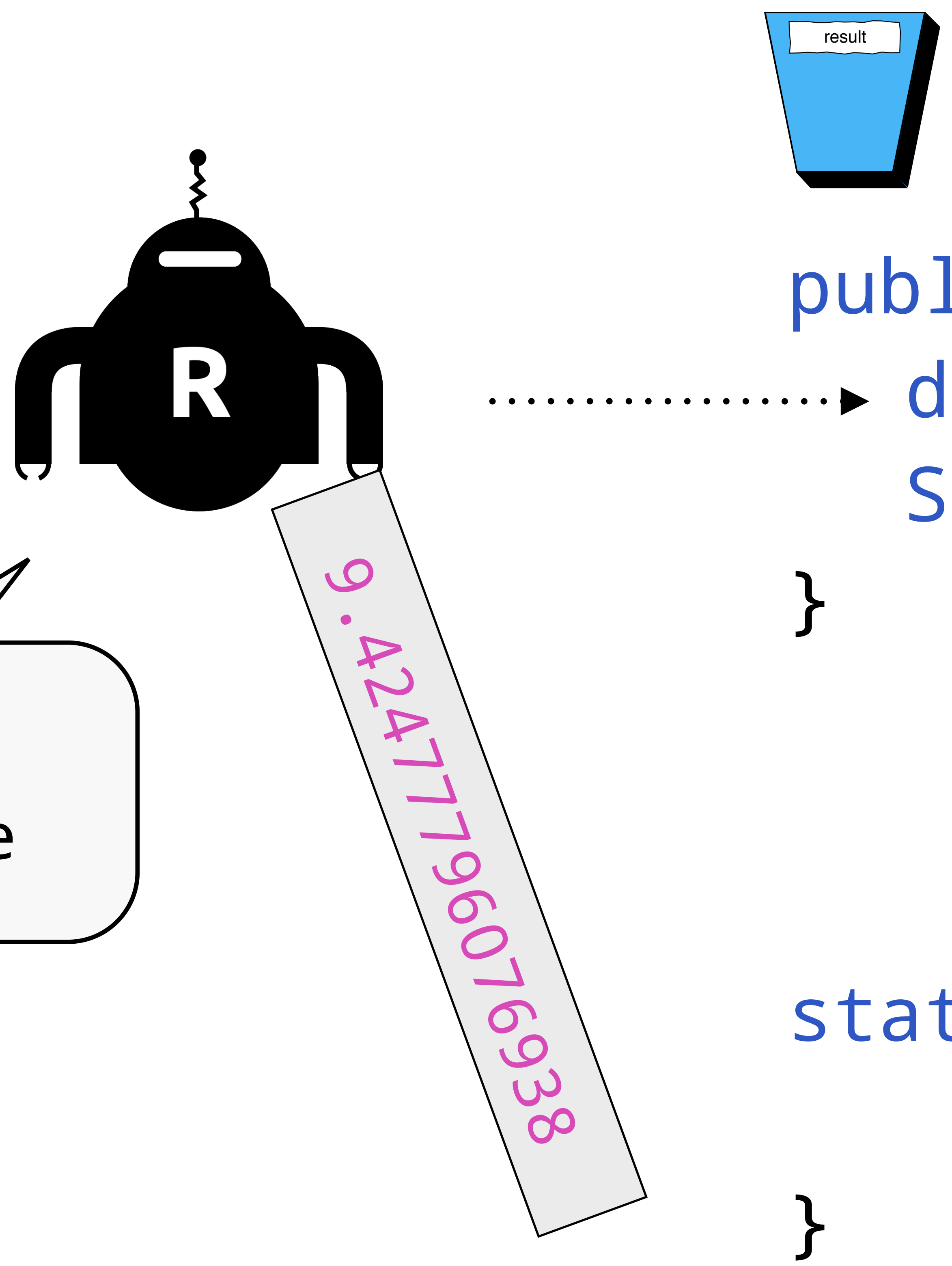
```
static double circumference( double radius ){
    return 9.42477796076938;
}
```



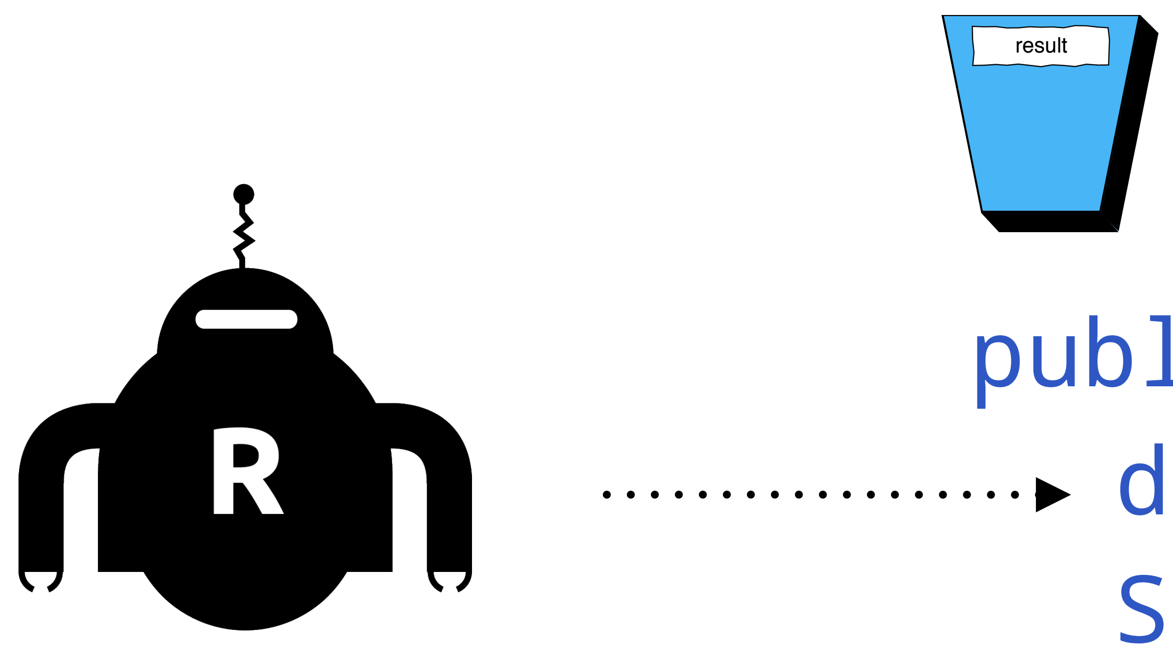
```
public static void main( String[] args ){
 ● double result = circumference( 1.5 );
   System.out.println( result );
}
```



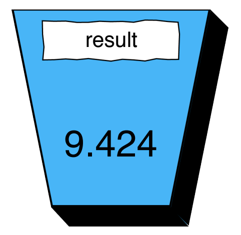
```
static double circumference( double radius ){
     radius * 2 * Math.PI;
}
```

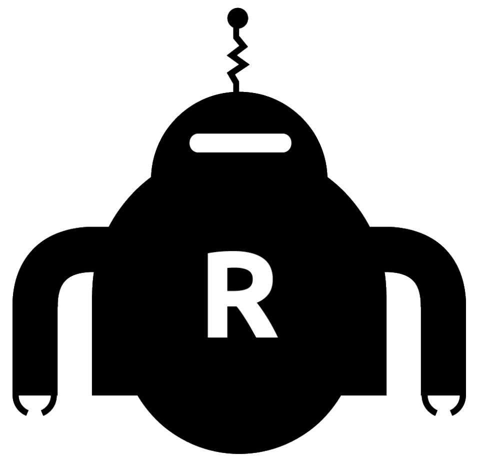


```
public static void main( String[] args ){
  double result = circumference( 1.5 );
  System.out.println( result );
}


static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```



```
public static void main( String[] args ){
  double result = 9.42477796076938;
  System.out.println( result );
}


static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```





```
public static void main( String[] args ){
   double result = 9.42477796076938;
   System.out.println( result );
}
```

```
static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

Console

```
9.42477796076938
```

```
public static void main( String[] args ){
   double result = circumference( 1.5 );
   System.out.println( result );
}
```

```
static double circumference( double radius ){
    return radius * 2 * Math.PI;
}
```

Console

```
9.42477796076938
```

Method overloading: Method that share name but have different number (or types) of arguments.

```
static void freeze(){
    // Freeze for a default duration.
}

static void freeze( double duration ){
    // Freeze for a specified duration.
}
```

Documenting methods: Use @param and @return.

```java
/**
 * This method will place birds in the world.
 * @param count The number of birds to place.
 * @param radius The radius in which the spread out the birds.
 * @return A boolean flag indicating the success of the operation.
 */
static boolean placeBirds( int count, double radius ){
    return false; // Missing implementation.
}
```