Methods Calling

At a fall fair.

The Formal Specification for Method Calls from the Official Java Documentation

15.12. Method Invocation Expressions

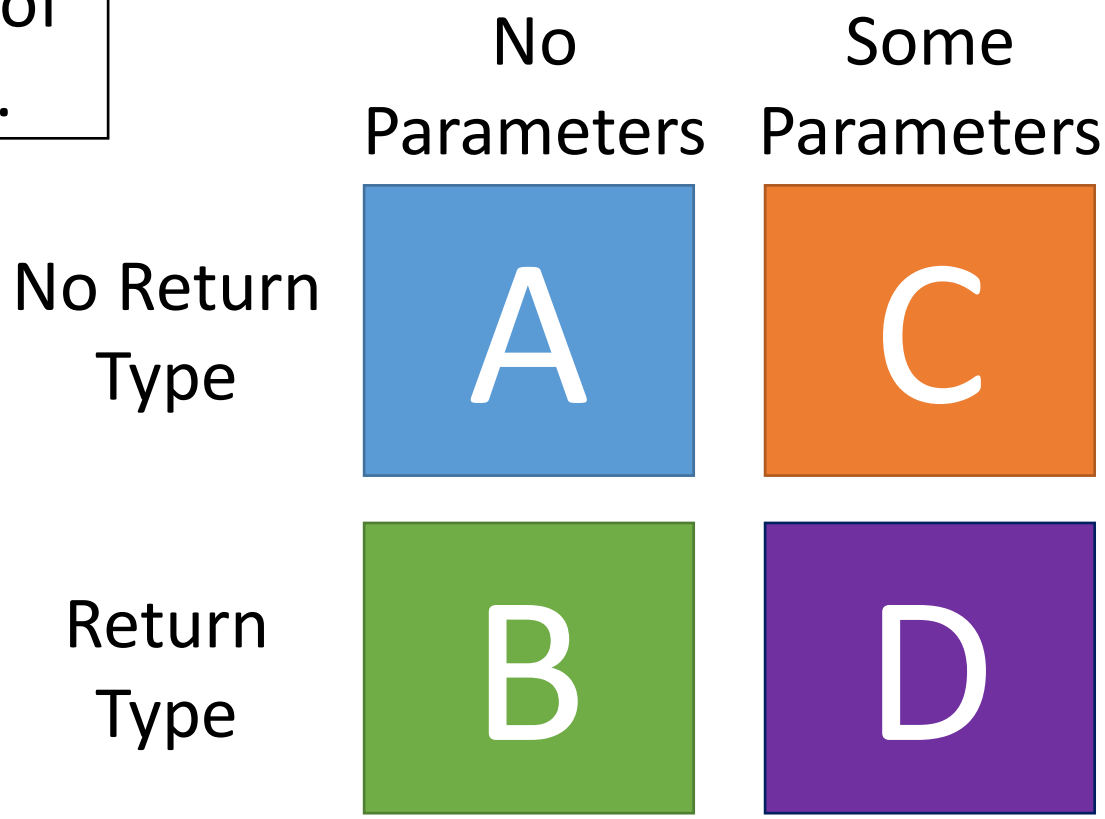
A method invocation expression is used to invoke a class or instance method.

```
MethodInvocation:
    MethodName ( ArgumentList_opt )
    Primary . NonWildTypeArguments_opt Identifier ( ArgumentList_opt )
    super . NonWildTypeArguments_opt Identifier ( ArgumentList_opt )
    ClassName . super . NonWildTypeArguments_opt Identifier ( ArgumentList_opt )
    TypeName . NonWildTypeArguments Identifier ( ArgumentList_opt )
```

The definition of ArgumentList from §15.9 is repeated here for convenience:

```
ArgumentList:
    Expression
    ArgumentList , Expression
```

I normally break it down into 4 types of methods….



Use this method signature to classify this method.

	No Parameters	Some Parameters
No Return Type	A	C
Return Type	B	D

```
public void tulip()
```

Use this method signature to classify this method.

	No Parameters	Some Parameters
No Return Type	A	C
Return Type	B	D

```
public int daffodil()
```

Use this method signature to classify this method.

	No Parameters	Some Parameters
No Return Type	A	C
Return Type	B	D

```
public double snowdrop(int bulb)
```

Use this method signature to classify this method.

	No Parameters	Some Parameters
No Return Type	A	C
Return Type	B	D

```
public void hyacinth(char a)
```

Use this method signature to classify this method.

	No Parameters	Some Parameters
No Return Type	A	C
Return Type	B	D

```
public boolean trillium()
```

Try question 1 now.

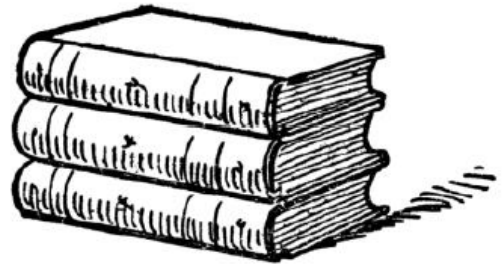
Method Call Practice

2.3K

Name: _____

Method calls fit into one of these four categories:

	No Parameters	Parameters
No Return Type	A (easiest)	C
Return Type	B	D (hardest)



[From PowerPoint]

1. Identify the category (A, B, C, D) for each of these method signatures.

____ (a) public void tulip ()

____ (b) public int daffodil ()

____ (c) public double snowdrop (int bulb)

____ (d) public void hyacinth(char a)

____ (e) public Boolean trillium ()

Solutions.
Did you get it right?

[From PowerPoint]

1. Identify the category (A, B, C, D) for each of these method signatures.

A (a) public void tulip ()

B (b) public int daffodil ()

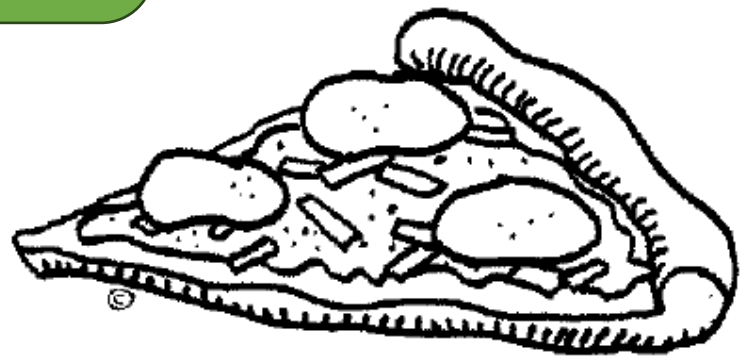
D (c) public double snowdrop (int bulb)

C (d) public void hyacinth(char a)

B (e) public Boolean trillium ()

Try question 3 & 4 now.

3. Why is a method call needed? [Circle the best answer]
   (a) A method call is used to initialize the screen.
   (b) It runs the method's code.
   (c) It uses the method signature.
   (d) It helps to black box test a method.



4. This was the method signature from the rollBig method in the Pizza Party app.

```
public void rollbig(JButton square)
```

   (a) How did will call it for the "f" button? _____

   (b) What type of method was it? (A, B, C, D) _____

   (c) What was its parameter type? _____

Solutions.
Did you get it right?

3. Why is a method call needed? [Circle the best answer]
   (a) A method call is used to initialize the screen.
   (b) It runs the method's code.
   (c) It uses the method signature.
   (d) It helps to black box test a method.

Solutions.
Did you get it right?

4. This was the method signature from the rollBig method in the Pizza Party app.

```
public void rollbig(JButton square)
```

(a) How did will call it for the "f" button? rollbig (f);

(b) What type of method was it? (A, B, C, D) C

(c) What was its parameter type? JButton

The way you call a method boils down to this:

Colour Code:

Always

If Parameters

If Return Type

```
parameterType parameter = value;
returnType variable = methodName (parameter);
```

parameterType parameter = value;

returnType variable = **methodName** (parameter);

5. Which piece of the method call are used by each type of method? Highlight the piece if it is needed.

A	`public void parrot()`	parameterType parameter = value; returnType variable = **methodName** (parameter **);**
B	`public char ape()`	parameterType parameter = value; returnType variable = **methodName** (parameter **);**
C	`public void fish(int guppy)`	parameterType parameter = value; returnType variable = **methodName** (parameter **);**
D	`public String rat (char cheese)`	parameterType parameter = value; returnType variable = **methodName** (parameter **);**

parameterType parameter = value;

returnType variable = **methodName** (parameter);

5. Which piece of the method call are used by each type of method? Highlight the piece if it is needed.

A	`public void parrot()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
B	`public char ape()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
C	`public void fish(int guppy)`	parameterType parameter = value; returnType variable = **methodName** (parameter);
D	`public String rat (char cheese)`	parameterType parameter = value; returnType variable = **methodName** (parameter);

parameterType parameter = value;

returnType variable = **methodName** (parameter);

5. Which piece of the method call are used by each type of method? Highlight the piece if it is needed.

A	`public void parrot()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
B	`public char ape()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
C	`public void fish(int guppy)`	parameterType parameter = value; returnType variable = **methodName** (parameter);
D	`public String rat (char cheese)`	parameterType parameter = value; returnType variable = **methodName** (parameter);

parameterType parameter = value;

returnType variable = **methodName** (parameter);

5. Which piece of the method call are used by each type of method? Highlight the piece if it is needed.

A	`public void parrot()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
B	`public char ape()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
C	`public void fish(int guppy)`	parameterType parameter = value; returnType variable = **methodName** (parameter);
D	`public String rat (char cheese)`	parameterType parameter = value; returnType variable = **methodName** (parameter);

parameterType parameter = value;

returnType variable = **methodName** (parameter);

5. Which piece of the method call are used by each type of method? Highlight the piece if it is needed.

A	`public void parrot()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
B	`public char ape()`	parameterType parameter = value; returnType variable = **methodName** (parameter);
C	`public void fish(int guppy)`	parameterType parameter = value; returnType variable = **methodName** (parameter);
D	`public String rat (char cheese)`	parameterType parameter = value; returnType variable = **methodName** (parameter);

Call this method:

```
public boolean cat(int dog)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

Call this method:

```
public boolean cat(int dog)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

Call this method:

```
public boolean cat(int dog)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
    int dog = 5;
returnType variable = methodName (parameter);
```

Call this method:

```
public boolean cat(int dog)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
```

```
    int dog = 5;
```

```
returnType variable = methodName (parameter);
```

```
    boolean ans = cat(dog);
```

1

Call this method:

```
public void parrot()
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

1

Call this method:

```
public void parrot()
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
        --not needed--
returnType variable = methodName (parameter);
        parrot();
```

2

Call this method:

```
public void fish(char guppy)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

2

Call this method:

```
public void fish(char guppy)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
    char guppy = 'a';
returnType variable = methodName (parameter);
    fish(guppy);
```

3

Call this method:

```
public char ape()
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

3

Call this method:

```
public char ape()
```

1. Check off the pieces you have:

Colour Code:

- ✓ Always
- If Parameters
- ✓ If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
        --not needed--
returnType variable = methodName (parameter);
        char ans = ape();
```

4

Call this method:

```
public String rat(char cheese)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

4

Call this method:

```
public String rat(char cheese)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
```

```
char cheese = 'j';
```

```
returnType variable = methodName (parameter);
```

```
String ans = rat(cheese);
```

5

Call this method:

```
public void okapi(JButton b)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

5

Call this method:

```
public void okapi(JButton b)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
JButton b = new JButton("hi");

returnType variable = methodName (parameter);
okapi(b);
```

Fill in question 6(a) now

6. Call these methods. [From PowerPoint]

(a) `public void okapi(JButton b)`

JButton	b	=	new JButton("hi");
Param Type	Param Name	=	Initial Value;

			okapi	(b);
Return Type	Variable Name	=	Method Name	(Param Name);

6

Call this method:

```
public double hamster()
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

6

Call this method:

```
public double hamster()
```

1. Check off the pieces you have:

Colour Code:

- ✓ Always
- If Parameters
- ✓ If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
        --not needed--
returnType variable = methodName (parameter);
        double ans = hamster();
```

Fill in question 6(b) now

```
(b) public double hamster()
```

Param Type	Param Name	=	Initial Value;
double	ans	=	hamster();
Return Type	Variable Name	=	Method Name (Param Name);

7

Call this method:

```
public int lion(double rhino)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

7

Call this method:

```
public int lion(double rhino)
```

1. Check off the pieces you have:

Colour Code:

- Always
- If Parameters
- If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;
```

```
double rhino = Double.parseDouble(in.getText().toString());
```

```
returnType variable = methodName (parameter);
```

```
int ans = lion(rhino);
```

Fill in question 6(c) now

```
(c) public int lion(double rhino)
```

double rhino = 3.14159;

Param Type Param Name = Initial Value;

int ans = lion(rhino);

Return Type Variable Name = Method Name (Param Name);

There are many variations of method calls.

The template does not need to be used exactly.

Other ways that you can call methods with return types:

```
public boolean win()
```

Directly in a boolean expression:

```
if (win())
    result.setText("Good job");
```

```
public String format()
```

Directly outputted to the screen:

```
result.setText("Date: "+format());
```

Other ways that you can call methods with parameters:

```
public void switchTurn(ImageView turn)
```

The parameter can be an ImageView.

```
ImageView turn = (ImageView) findViewById(R.id.turn);
switchTurn(turn);
```

```
public void format(String s)
```

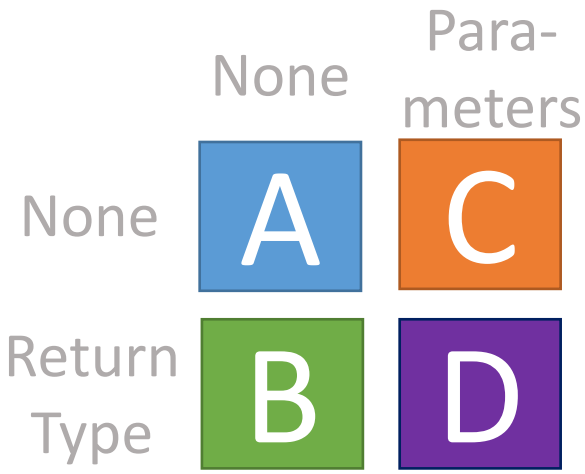
The parameter can have a different name than the variable.

```
String name = input.getText().toString();
format(name);
```

The data of the right type can be sent in directly.

```
format("Ms. Gorski");
```

Classify the following methods:




```
1. public void tree()
2. public String maple()
3. public void oak (String leaf)
4. public char pine (int needles)
5. public void spruce (boolean bird)
6. public boolean isTree()
7. public void chestnut()
8. public void cherry(ImageView i)
9. public int crabApple(char rotten, TextView t)
```

1

Call this method:

```
public void tree()
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

2

Call this method:

```
public String maple()
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

3

Call this method:

```
public void oak(String leaf)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

4

Call this method:

```
public char pine(int needles)
```

1. Check off the pieces you have:

Colour Code:

Always

If Parameters

If Return Type

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

5

Call this method:

1. Check off the pieces:

Colour Code:

Always

If Parameters

If Return Type

```
public void cherry(ImageView apple)
```

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

6

Call this method:

1. Check off the pieces:

Colour Code:

Always

If Parameters

If Return Type

```
public void oak(ImageView acorn, int leaf)
The ImageView's id is small, the int is stored in big
```

2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

7

Call this method:

1. Check off the pieces:

Colour Code:

Always

If Parameters

If Return Type

```
public void pine(Button needle, char cone)
The Button's id is spruce, the char is named squirrel
```

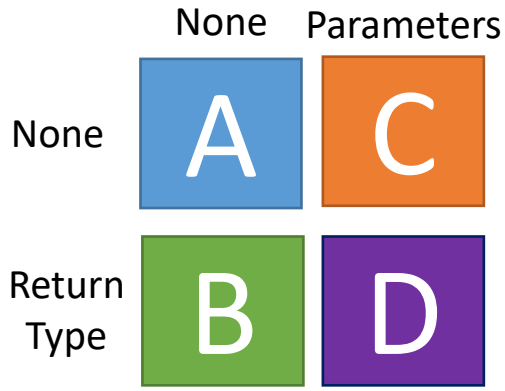
2. Use the template to build your method call:

```
parameterType parameter = value;

returnType variable = methodName (parameter);
```

parameterType parameter = value;

returnType variable = methodName (parameter);




```
1. public void tree()
2. public String maple()
3. public void oak (String leaf)
4. public char pine (int needles)
5. public void spruce (boolean bird)
6. public boolean isTree()
7. public void chestnut()
8. public void cherry(ImageView i)
9. public int crabApple(char rotten, TextView t)
```

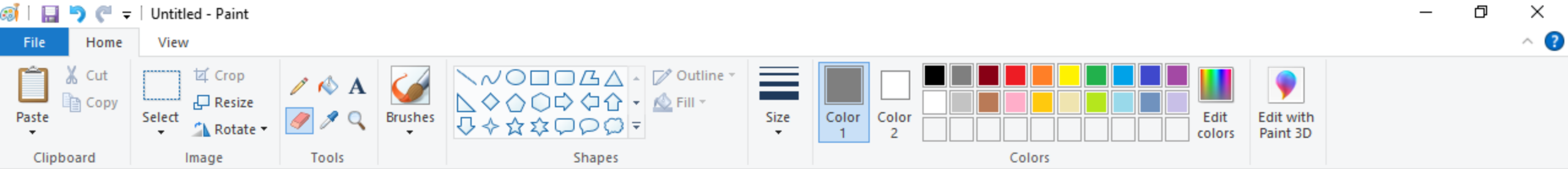
```
paramType parameter = IO.input[paramType](“prompt”);
returnType variable = s.methodName (parameter);
System.out.println(“That is: ” +variable);

paramType parameter = IO.input[paramType](“prompt”);
System.out.println(“That is: ” + s.methodName (paramValue));

returnType variable = s.methodName (paramValue);
System.out.println(“That is: ” +variable);

System.out.println(“That is: ” + s.methodName (paramValue));
```

```
parameterType parameter = value;
returnType variable = methodName ( parameter );
```



Input: for parameters. One each.

```
paramType parameter = IO.input[paramType](“prompt”);
returnType variable = s.methodName (parameter);
System.out.println(“That is: ” +variable);
```

Output: for return type.

Input: for parameters. One each.

```
paramType parameter = IO.input[paramType](“prompt”);
System.out.println(“That is: ” + s.methodName (paramValue));
```

Output: for return type.

```
returnType variable = s.methodName (paramValue);
```

Output: for return type.

Input: for parameters.

```
System.out.println(“That is: ” + s.methodName (paramValue));
```

Output: for return type.

Input: for parameters.