Input and Output

Parameters and Return Types in Methods

```
public double queenOfHearts (char suite) {
   if (suit == 'h')
      return 3.14159;
   else
      return 1.99999;
}

public int whiteRabbit (int watch) {
   return watch++;
}

public void madHatter (int teacup) {
    return teacup--;
}

public void cheshireCat () {
    System.out.println(" Grinning cat! ");
}
```

How many methods?

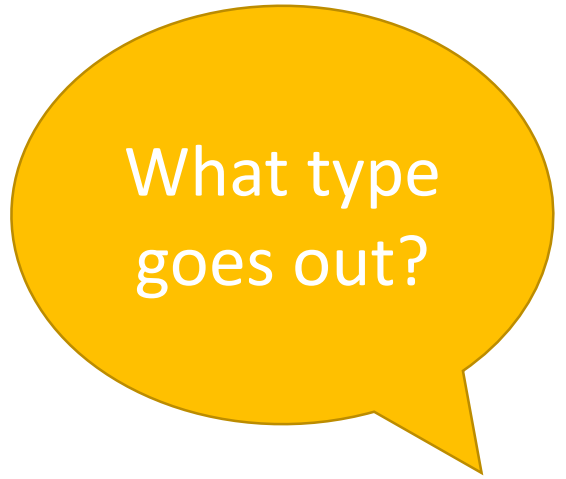
What are their names?

```
public double queenOfHearts (char suite) {
   if (suit == 'h')
      return 3.14159;
   else
      return 1.99999;
}

public int whiteRabbit (int watch) {
   return watch++;
}

public void madHatter (int teacup) {
    return teacup--;
}

public void cheshireCat () {
    System.out.println(" Grinning cat! ");
}
```



```
public double queenOfHearts (char suite) {
   if (suit == 'h')
      return 3.14159;
   else
      return 1.99999;
}

public int whiteRabbit (int watch) {
   return watch++;
}

public void madHatter (int teacup) {
    teacup--;
}

public void cheshireCat () {
    System.out.println(" Grinning cat! ");
}
```

Where are the return lines?

What is the parameter name?

Where is the method signature?

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

```
__________  _______________ _______________(________ _________{
```

start word *return type (out)* *method name* *param type* *name (in)*

```
  if(_____________.equalsIgnoreCase(“paper”))
      return _________________;
  else if (_____________.equalsIgnoreCase(“p”))
      return _________________;
  else
      return _________________;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

```
public _____ _____ _____(_____ _____{
```

start word *return type (out)* *method name* *param type* *name (in)*

```
    if(_____.equalsIgnoreCase(“paper”))
        return _____;
    else if (_____.equalsIgnoreCase(“p”))
        return _____;
    else
        return _____;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

public boolean _____ (_____ _____ {

start word *return type (out)* *method name* *param type* *name (in)*

```
    if(_____.equalsIgnoreCase(“paper”))
        return _____;
    else if (_____.equalsIgnoreCase(“p”))
        return _____;
    else
        return _____;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

public boolean wantPaper (_____ _____ {

start word *return type (out)* *method name* *param type* *name (in)*

```
    if(_____.equalsIgnoreCase(“paper”))
        return _____;
    else if (_____.equalsIgnoreCase(“p”))
        return _____;
    else
        return _____;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

public _____ boolean _____ wantPaper _____ (String _____ word _____ {

start word *return type (out)* *method name* *param type* *name (in)*

```
    if(_____.equalsIgnoreCase(“paper”))
        return _____;
    else if (_____.equalsIgnoreCase(“p”))
        return _____;
    else
        return _____;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

```
public boolean wantPaper (String word ) {
```

start word *return type (out)* *method name* *param type* *name (in)*

```
    if( word .equalsIgnoreCase(“paper”))
        return true ;
    else if (_____.equalsIgnoreCase(“p”))
        return _____;
    else
        return _____;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

```
public boolean wantPaper(String word){
```

start word *return type (out)* *method name* *param type* *name (in)*

```
    if(word.equalsIgnoreCase(“paper”))
        return true;
    else if (word.equalsIgnoreCase(“p”))
        return true;
    else
        return ____________________;
}
```

Comes in:

- A word

Goes out:

- True if it is a word that indicates the user wants “paper”
- False otherwise

public boolean wantPaper (String word) {

start word *return type (out)* *method name* *param type* *name (in)*

```
    if( word .equalsIgnoreCase(“paper”))
        return true ;
    else if ( word .equalsIgnoreCase(“p”))
        return true ;
    else
        return false ;
}
```

Comes in:

- An ending number for a series
- What number to skip count by

Goes out:

- Nothing
- A series of numbers starting at 0 and going by the amount appear

_____ _____ _____(_____ _____, _____ _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    for (int i = 0; i< _____; i+=_____) {
       System.out.print(i + " ");
    System.out.println();
 }
```

Comes in:

- An ending number for a series
- What number to skip count by

Goes out:

- Nothing
- A series of numbers starting at 0 and going by the amount appear

public _____ _____ _____(_____ _____, _____ _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    for (int i = 0; i< _____; i+=_____) {
        System.out.print(i + " ");
    System.out.println();
}
```

Comes in:

- An ending number for a series
- What number to skip count by

Goes out:

- Nothing
- A series of numbers starting at 0 and going by the amount appear

public void _____ _____ _____(_____ _____, _____ _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    for (int i = 0; i< _____; i+=_____) {
        System.out.print(i + " ");
    System.out.println();
}
```

Comes in:

- An ending number for a series
- What number to skip count by

Goes out:

- Nothing
- A series of numbers starting at 0 and going by the amount appear

public _____ void _____ skipCount _____(_____ _____, _____ _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    for (int i = 0; i< _____; i+=_____) {
        System.out.print(i + " ");
    System.out.println();
}
```

Comes in:

- An ending number for a series
- What number to skip count by

Goes out:

- Nothing
- A series of numbers starting at 0 and going by the amount appear

public _____ void _____ skipCount _____ (int _____ end _____ , int _____ skip _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    for (int i = 0; i< ____________; i+=_____) {
        System.out.print(i + " ");
    System.out.println();
}
```

Comes in:

- An ending number for a series
- What number to skip count by

Goes out:

- Nothing
- A series of numbers starting at 0 and going by the amount appear

public _____ void _____ skipCount _____ (int _____ end _____ , int _____ skip _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    for (int i = 0; i< ____end____; i+=_skip_) {
        System.out.print(i + " ");
    System.out.println();
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

```
_____ _____ _____(_____ _____, _____ _____){
```

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    if( _____> _____)
        return “Too high”;
    else if ( _____< _____)
        return “Too high”;
    else
        return “_____”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

```
public _____ _____ _____(_____ _____, _____ _____){
```

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    if( _____> _____)
        return “Too high”;
    else if ( _____< _____)
        return “Too high”;
    else
        return “_____”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

```
public String _____(_____ _____, _____ _____){
```

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
   if( guess_____> _____)
        return “Too high”;
   else if ( _____< _____)
        return “Too high”;
   else
        return “_____”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

public String hint _____ _____ _____(_____ _____, _____ _____){

start word *return type (out)* *method name* *param type* *name (in)* , *param type* *name (in)*

```
    if( _____> _____)
        return “Too high”;
    else if ( _____< _____)
        return “Too high”;
    else
        return “_____”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

public _____ String _____ hint _____ (int _____ target _____ , int _____ guess _____) {

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    if( _____> _____)
        return “Too high”;
    else if ( _____< _____)
        return “Too high”;
    else
        return “_____”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

```
public  String  hint (int target, int guess) {
```

start word *return type (out)* *method name* *param type* *name (in)* *param type* *name (in)*

```
    if( guess > target )
        return “Too high”;
    else if ( ____________ < ____________ )
        return “Too high”;
    else
        return “_____________________”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

```
public String hint(int target, int guess){
```

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    if( guess > target )
        return “Too high”;
    else if ( guess < target )
        return “Too high”;
    else
        return “_____”;
}
```

Comes in:

- A target number between 1 and 100
- A number trying to guess the target

Goes out:

- Words indicating if the guess is “Too high”, “Too low” or “Just right”

```
public String hint(int target, int guess){
```

start word *return type (out)* *method name* *param type* *name (in)* *, param type* *name (in)*

```
    if( guess > target )
        return “Too high”;
    else if ( guess < target )
        return “Too high”;
    else
        return “Just right”;
}
```

Comes in:

- A maximum number

Goes out:

- Nothing is returned
- However, the fizz-buzz numbers from 0 to the maximum are printed


```
_____ _____ _____(_____ _____) {
```

start word *return type (out)* *method name* *param type* *name (in)*

```
   for (int i = 0; i< _____; i++) {
      if (i % 15 == 0)
         System.out.println("Fizz-Buzz");
      else if (i % 3 == 0)
         System.out.println("Fizz");
      else if (i % 5 == 0)
         System.out.println("Buzz");
      else
         System.out.println(i+"");
}
```

Comes in:

- A maximum number

Goes out:

- Nothing is returned
- However, the fizz-buzz numbers from 0 to the maximum are printed

public _____ _____ _____(_____ _____) {

start word *return type (out)* *method name* *param type* *name (in)*

```
for (int i = 0; i< _____; i++) {
    if (i % 15 == 0)
        System.out.println("Fizz-Buzz");
    else if (i % 3 == 0)
        System.out.println("Fizz");
    else if (i % 5 == 0)
        System.out.println("Buzz");
    else
        System.out.println(i+"");
}
```

Comes in:

- A maximum number

Goes out:

- Nothing is returned
- However, the fizz-buzz numbers from 0 to the maximum are printed

public void _____ _____(_____ _____) {

start word *return type (out)* *method name* *param type* *name (in)*

```
for (int i = 0; i< _____; i++) {
    if (i % 15 == 0)
        System.out.println("Fizz-Buzz");
    else if (i % 3 == 0)
        System.out.println("Fizz");
    else if (i % 5 == 0)
        System.out.println("Buzz");
    else
        System.out.println(i+"");
}
```

Comes in:

- A maximum number

Goes out:

- Nothing is returned
- However, the fizz-buzz numbers from 0 to the maximum are printed

public _____ void _____ printFizzBuzz _____ (_____ _____) {

start word *return type (out)* *method name* *param type* *name (in)*

```
for (int i = 0; i< _____; i++) {
    if (i % 15 == 0)
        System.out.println("Fizz-Buzz");
    else if (i % 3 == 0)
        System.out.println("Fizz");
    else if (i % 5 == 0)
        System.out.println("Buzz");
    else
        System.out.println(i+"");
}
```

Comes in:

- A maximum number

Goes out:

- Nothing is returned
- However, the fizz-buzz numbers from 0 to the maximum are printed

public _____ void _____ printFizzBuzz _____ (int _____ max _____) {

start word *return type (out)* *method name* *param type* *name (in)*

```
for (int i = 0; i< max ; i++) {
    if (i % 15 == 0)
        System.out.println("Fizz-Buzz");
    else if (i % 3 == 0)
        System.out.println("Fizz");
    else if (i % 5 == 0)
        System.out.println("Buzz");
    else
        System.out.println(i+"");
}
```

Comes in:

- A maximum number

Goes out:

- Nothing is returned
- However, the fizz-buzz numbers from 0 to the maximum are printed

public _____ void _____ printFizzBuzz _____ (int _____ max _____) {

start word *return type (out)* *method name* *param type* *name (in)*

```
for (int i = 0; i< ___________; i++) {
    if (i % 15 == 0)
        System.out.println("Fizz-Buzz");
    else if (i % 3 == 0)
        System.out.println("Fizz");
    else if (i % 5 == 0)
        System.out.println("Buzz");
    else
        System.out.println(i+"");
}
```