Chapter 14

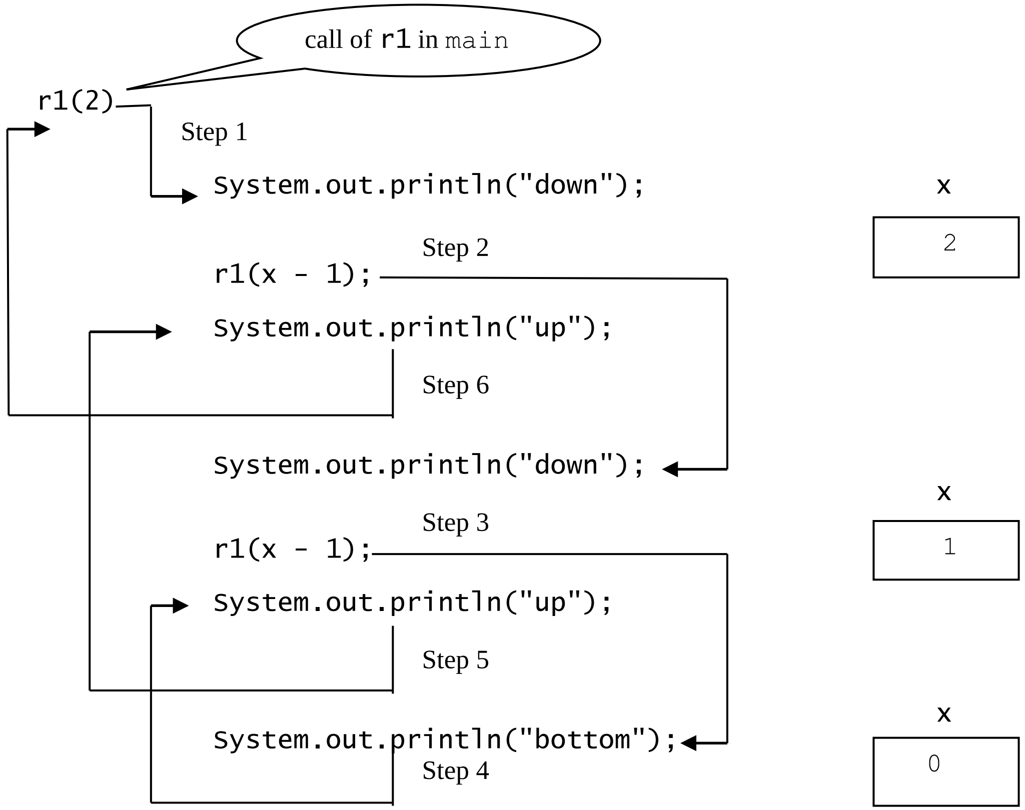
Recursive Methods

You already understand recursive methods

1) The parameters, if any, for the called method are created.
2) The values of the arguments, if any, in the call are automatically assigned to their corresponding parameters in the called method.
3) Local variables, if any, are created.
4) The body of the called method is executed.
5) Local variables and parameters created in steps 1 and 3 are destroyed.
6) The called method returns to its caller.

Stepping through a recursive method

```
class Recursion1
{
   public static void main(String[] args)
   {
      r1(2);
   }
  //----------------------------------
   public static void r1(int x)
   {
      if (x == 0)
         System.out.println("bottom");
      else
      {
         System.out.println("down");
         r1(x - 1);
         System.out.println("up");
      }
   }
}
```



```
r1(2)

System.out.println("down");
r1(x - 1);
System.out.println("up");

System.out.println("down");
r1(x - 1);
System.out.println("up");

System.out.println("bottom");
```

An easy way to determine what a recursive method does

$\left.\texttt{bottom}\right\}$ what `r1` displays when it is passed `0`

```
     System.out.println("down");
     r1(x - 1);
     System.out.println("up");
```

$\left.\begin{array}{l}\texttt{down}\\\texttt{bottom}\\\texttt{up}\end{array}\right\}$ What `r1` displays when it is passed `1`

```
System.out.println("down");
r1(x - 1);
System.out.println("up");
```

```
down
down
bottom
up
up
```

$\Bigg\}$ what `r1` displays when it is passed `2`

```java
class Recursion2
{
   public static void main(String[] args)
   {
      r2(5);
      System.out.println(); // go to next line
   }
   //-----------------------------------
   public static void r2(int x)
   {
      if (x == 0)
         System.out.print("E");
      else
      if (x == 1)
      {
         System.out.print("A");
         r2(6);   // parameter value jumps up to 6
         System.out.print("B");
      }
      else
      {

         System.out.print("C");
         r2(x - 2);
         System.out.print("D");

      }
   }
}
```

Reverse trace

5 → 3 → 1 → 6 → 4 → 2 → 0

x	r2 displays
0	E
2	CED
4	CCEDD
6	CCCEDDD
1	ACCCEDDDB
3	CACCCEDDDBD
5	CCACCCEDDDBDD

Performing a sub-task using a recursive call

```java
class Recursion3
{
   public static void main(String[] args)
   {
      numbersToOne(10);
   }
   //---------------------------------
   public static void numbersToOne(int n)
   {
      if (n > 0)
      {
         System.out.println(n);
         numbersToOne(n - 1);
      }
   }
}
```

Recursive `reverse`

```
                    tail
                     ↓
        --------------------------
reverse(s.substring(1, s.length())) + s.substring(0,1)
----------------------------------   ----------------
                     ↑                          ↑
          tail reversed                 first character
          by recursive call
```

```java
class Recursion4
{
   public static void main(String[] args)
   {
       System.out.println(reverse(""));
       System.out.println(reverse("A"));
       System.out.println(reverse("ABCDEF"));
   }
   //---------------------------------
   public static String reverse(String s)
   {
      if (s.length() <= 1)
         return s;
      return
         reverse(s.substring(1, s.length()))
                    + s.substring(0, 1);
   }
}
```

Loops versus recursion

```
for (int i = n; i >= 1; i--)
    System.out.println(i);
```