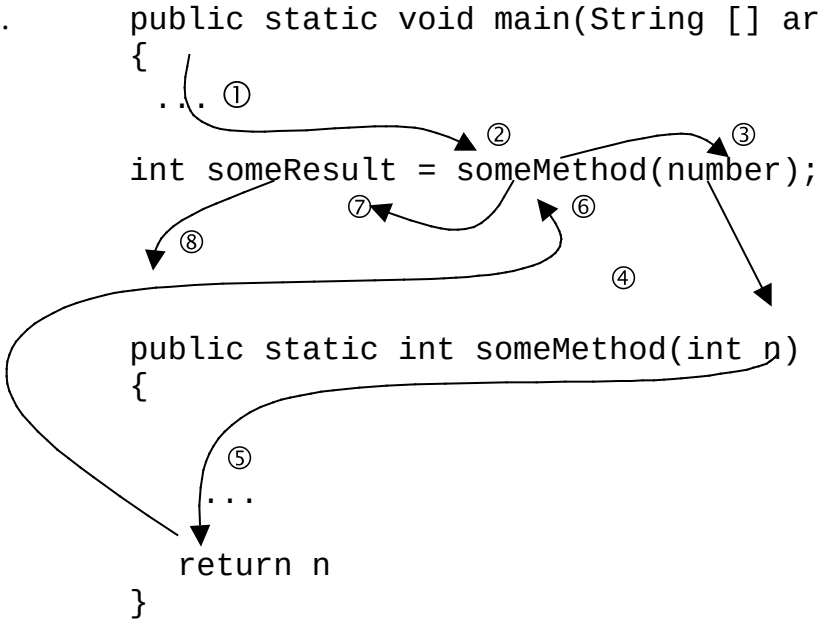
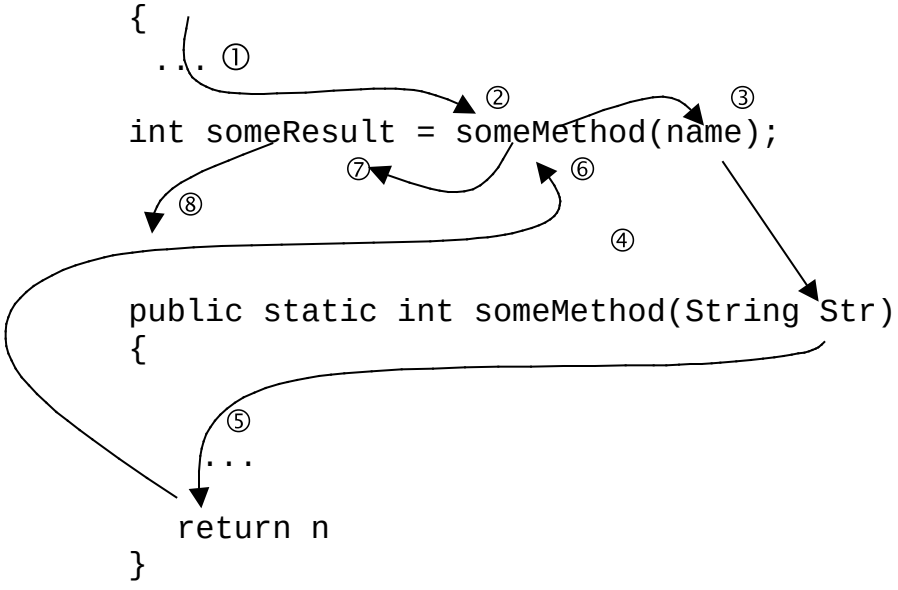
Exercises 4.2

1.

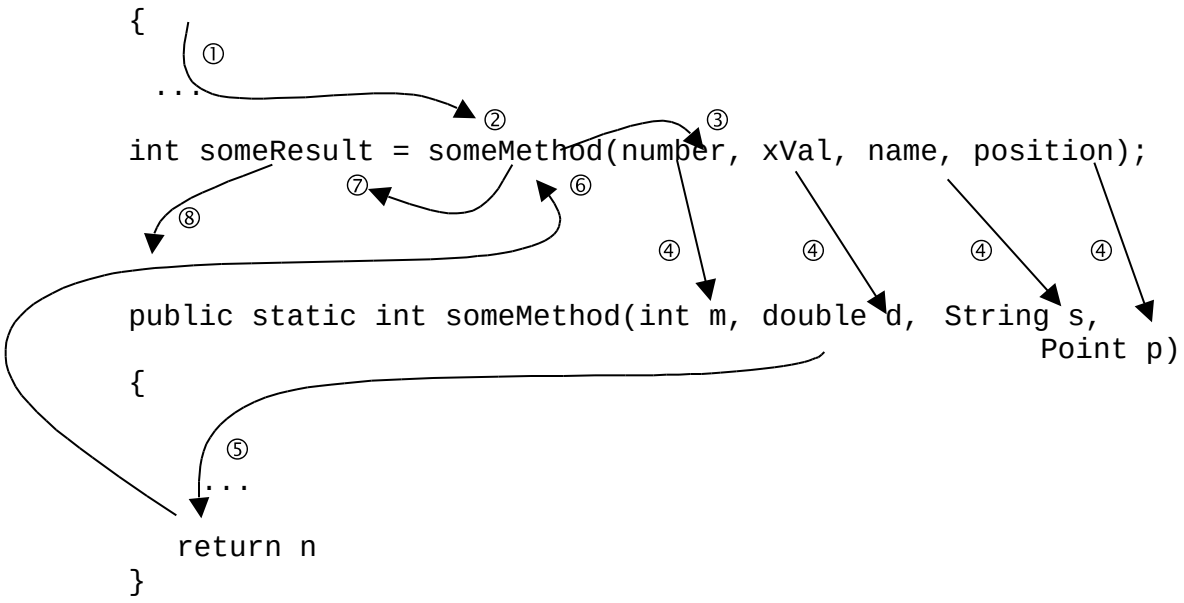


2.



3.

```
public static void main(String [] args)
```



4.

```
/** range(int1, int2) receives two integers and
 *    returns the range between them.
 *
 *  Receive: int1, int2, integers
 *  Return:  the range between int1 and int2
 */

public static int range(int int1, int int2)
{
  return Math.abs(int1 - int2);
}
```

5.

```
/** wages(hoursWorked, hourlyPayRate)calculates wages earned
 *    given hours worked and hourly pay rate.
 *
 *  Receive: hoursWorked, the hours worked;
 *           hourlyPayRate, the hourly pay rate
 *  Return:  the wages earned
 */
public static double wages(double hoursWorked, double hourlyPayRate)
{
  return hoursWorked * hourlyPayRate;
}
```

6.

```
/** circleCircumference(radius) returns the circumference
 *     of a circle of given radius.
 *
 *  Receive: radius, the radius of a circle
 *  Return:  the circumference of the circle
 */

public static double circleCircumference(double radius)
{
   return 2.0 * Math.PI * radius;
}
```

7.

```
/** circleArea(radius) returns the area of a circle
 *    of given radius.
 *
 * Receive: radius, the radius of a circle
 * Return:  the area of the circle
 */

public static double circleArea(double radius)
{
   return Math.PI * radius * radius;
}
```

8.

```
/** rectanglePerimeter(height, width) returns the perimeter
 *    of a rectangle of given height and width.
 *
 *  Receive: height and width of the rectangle
 *  Return:  the perimeter of the rectangle
 */

public static double rectanglePerimeter(double height, double width)
{
   return 2.0 * (height + width);
}
```

9.

```
/** rectangleArea(height, width) returns the area
 *    of a rectangle of given height and width.
 *
 *  Receive: height and width of the rectangle
 *  Return:  the area of the rectangle
 */

public static double rectangleArea(double height, double width)
{
   return height * width;
}
```

10.

```
/* trianglePerimeter(side1, side2, side3) returns the perimeter
 *    of a triangle of given side lengths.
 *
 * Receive: side1, side2, side3, the lengths of the sides
 *          of the triangle
 * Return:  the perimeter of the triangle
 */

 public static double trianglePerimeter(double side1,
                                        double side2, double side3)
 {
   return side1 + side2 + side3;
 }
```

11.

```
/** triangleArea(side1, side2, side3) returns the area of a
 *    triangle of given side lengths.
 *
 * Receive: side1, side2, side3, the lengths of the sides
 *          of the triangle
 * Return:  the area of the triangle
 */

public static double triangleArea(double side1,
                                  double side2, double side3)
{
   double s = (side1 + side2 + side3) / 2.0;

   return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
}
```

12.

```
/** bacteriaPopulation(initialPop, time, k) returns the
 *    population of a colony of bacteria, given the
 *    initial population, the time elapsed, and the
 *    constant of growth of the colony.
 *
 * Receive: initialPop, the initial population of the colony
 *          time, the time elapsed
 *          k, the rate constant for the growth of the colony
 * Return:  population of the colony of bacteria
 */

 public static long bacteriaPopulation(long initialPop,
                                       double time, double k)
 {
   return (long)(initialPop * Math.exp(time * k));
 }
```

13.

```
/** secondsToMinutes(seconds) converts Seconds to Minutes.
 *
 *  Receive: seconds, the given number of seconds
 *  Return:  minutes, the equivalent number of minutes
 */

public static double secondsToMinutes(long seconds)
{
   return seconds / 60.0;
}
```

14.

```
/** minutesToHours(minutes) converts Minutes to Hours.
 *
 *  Receive: minutes, a given number of minutes
 *  Return:  the equivalent number of hours
 */

public static double minutesToHours(double minutes)
{
   return minutes / 60.0;
}
```

15.

```
/** hoursToDays(hours) converts Hours to Days.
 *
 *  Receive: hours, the given number of hours
 *  Return:  the equivalent number of days
 */

public static double hoursToDays(double hours)
{
   return hours / 24.0;
}
```

16.

```
/** secondsToDays(seconds) converts Seconds to Days.
 *
 *  Receive: a given number of seconds
 *  Return:  the equivalent number of days
 */

public static double secondsToDays(long seconds)
{
   return hoursToDays( minutesToHours( secondsToMinutes(seconds) ) );
}
```

17.

```
/** phoneDisplay(sevenDigits) receives a 7-digit number and displays
 *   it as a phone  * number should appear (###-####)
 *
 * Receive: sevenDigits, an int 7 digits long.
 * Output:  the same number in phone number format.
 *
 * Assumes: access to Screen class from ann.easyio
 */

public static void phoneDisplay(int sevenDigits)
{
   Screen theScreen = new Screen();
   int lastFour = sevenDigits % 10000,
       firstThree = sevenDigits / 10000;
   theScreen.println(firstThree + "-" + lastFour);
}
```

18.

```
/** windChillIndex(t, v) will calculate and return the wind chill
 *   index for a given temperature in degrees Fahrenheit and
 *   wind speed.
 *
 *  Receive: t, a temperature in degrees Fahrenheit
 *           v, a wind speed in miles per hour
 *  Return:  the wind chill index for these values.
 */

public static double windChillIndex(double t, double v)
{
   return 91.4 - (0.474677 - 0.020425
           * v + 0.303107 * Math.sqrt(v) )
            * (91.4 - t);
}
```

19.

```
/** heatIndex(t, r) will calculate and return the heat index for
 *   a given temperature in degrees Fahrenheit and relative humidity.
 *
 *  Receive: t, a temperature in degrees Fahrenheit
 *           r, a relative humidity
 *  Return:  the heat index for these values.
 */

public static double heatIndex(double t, double r)
{
   return -42.379 + 2.04901523 * t  +  10.14333127 * r
          - 0.22475541 * t * r      -  (6.83783e-3) * t * t
          - (5.481717e-2) * r * r   +  (1.22874e-3) * t * t * r
          + (8.5282e-4) * t * r * r -  (1.99e-6) * t * t * r * r;
}
```

20.

```
/* printZero() displays the "stick number" 0.
 *
 * Output:  the stick figure for zero
 */

public static void printZero()
{
   Screen theScreen = new Screen();
   theScreen.println(" --- \n"
                   +  "|   |\n"
                   +  "|   |\n"
                   +  "|   |\n"
                   +  " --- \n");
}
```