1.

```
/**
 * Repeatedly asks the user for a positive integer until the user enters
 * one. Returns the positive integer.
 *
 * @param in
 *          the input stream
 * @param out
 *          the output stream
 * @return a positive integer entered by the user
 */
private static int getPositiveInteger(SimpleReader in, SimpleWriter out) {

    int posInt = 0;

    while (posInt <= 0) {
        out.print("Enter a positive integer: ");
        String isInt = in.nextLine();
        if (FormatChecker.canParseInt(isInt)) {
            posInt = Integer.parseInt(isInt);
        }
    }
    return posInt;
}
```

2.

Code	**State**
int n =1;	
	n=1
int i = 2;	

	n=1
	i=2

while (i < 5) {	

	n = 1
	i=2

n = n + i;
i = i + 1;	

	n = 3
	i=3

n = n + i;
i = i + 1;	

	n=6
	i=4

n = n + i;
i = i + 1;	

	n=10
	i=5

n = n + i;
i = i + 1;	

	n=15
	i=6

}	

	n=15
	i=6

Code	**State**
int i = 2;	
	i=2
double n = 1.0/2.0;	

	i=2
	n=.5

while(i<=5) {	

	i=~~2 3 4~~5
	n=~~.5 1.0 1.33~~1.58

n = n + 1.0 / i;	

	i=~~2 3 4~~ 5
	n=~~1.0 1.33 1.58~~1.78

i = i+1;	

	i=~~3 4 5~~ 6
	n=~~1.0 1.33 1.58~~1.78

}	

	i = 6
	n = 1.7833333

Code	**State**

double x = 1.0;
double y = 1.0;	

	x = 1.0
	y = 1.0

while (x<1.8) {	

	x = ~~1.0 1.5~~ 1.75
	y = ~~1.0 .5~~ .25

y = y / 2.0;	

	x = ~~1.0 1.5~~ 1.75
	y = ~~.5 .25~~ .125

x = x + y;	

	x = ~~1.5 1.75~~ 1.875
	y = ~~.5 .25~~ .125

}	

	x = 1.875
	y = .125

Code	State
int x = 3;	
	x=3
int y = 4;	

	x=3
	y=4

while (y > 0) {	

	x=~~3 4 5~~ 6
	y=~~4 3 2~~ 1

x = x + 1;	

	x=~~4 5 6~~ 7
	y=~~4 3 2~~ 1

y = y - 1;	

	x=~~4 5 6~~ 7
	y=~~3 2 1~~ 0

}	

	x=7
	y=0

3.

a)

```
int sum = 0;
for(int i = 2; i < 100; i = i + 2) {
        sum = sum + i;
}
```

```
        b)
int i = 1;
int sum = 0;
while(i < 11) {
            sum = i * i + sum;
            i++;
}

        c)
int sum = 0;
for(int i = 0; i < 21; i++) {
            sum += Math.pow(2,0);
}
        d)
int sum = 0;
while(a < b) {
            if(a % 2 != 0) {
                        sum += a;
            }
            a++;
}
        e)
int sum = 0;
int i = 0;
while(input > 0) {
            if(i % 2 == 0) {
            sum += input % 10;
            input = input / 10;
            }
            i++;
}
        f)
int length = String.valueOf(input).length();
int[] digits = new int[length];
int i = 1;
while(i < length + 1) {
            digits[length - i] = input % 10;
            input = input / 10;
            i++;
}
int sum = 0;
for(int j = 0; j < length; j++) {
            if(j % 2 == 0) {
                        sum += digits[j];
            }
}
```

4.

```
public static int greater(int a, int b)
public static double smallest(double a, double b, double c)
public static boolean isPrime(int a, int b)
public static boolean contains(String a, String b)
public static double newBalance(double initialBalance, double r, double years)
public static void newBalance(double initialBalance, double r, double years)
public static void printCalendar(int day, int month, int year)
public static String weekday(int day, int month, int year)
public static int randPosInt(int n)
```

5.

The method does not swap the contents of x and y because the arguments are copied to the formal parameters of the method, but the values at the end of the method are not copied back to x and y. In other words, x and y do not change their values just because they were used as arguments in falseSwap.

6.

```
/**
  * Tests if x, y, and z are all equal to each other.
  *
  * @param x
  *          any integer
  *
  * @param y
  *          any integer
  *
  * @param z
  *          any integer
  *
  * @return true if x, y, z are equal and false otherwise
  */
 private static boolean allTheSame(int x, int y, int z) {
     return x == y && y == z;
 }

 /**
  * Tests if x, y, and z are all different each other.
  *
  * @param x
  *          any integer
  *
  * @param y
  *          any integer
  *
  * @param z
  *          any integer
  *
  * @return true if x, y, z have no equalities and false otherwise
  */
 private static boolean allDifferent(int x, int y, int z) {
     return x != y && y != z && x != z;
 }

 /**
  * Tests if x, y, and z are all sorted in ascending order.
  *
  * @param x
  *          any integer
  *
  * @param y
  *           any integer
  *
  * @param z
  *           any integer
  *
  * @return true if x, y, z are in ascending order and false otherwise
  */
 private static boolean sorted(int x, int y, int z) {
     return x <= y && y <= z;
 }

 /**
  * Main method.
  *
  * @param args
  *           the command line arguments
  */
 public static void main(String[] args) {
     SimpleReader in = new SimpleReader1L();
     SimpleWriter out = new SimpleWriter1L();
     /*
      * Testing boolean methods
      */
     int a = 5, b = 6, c = 7;
     int a1 = -6, b1 = -6, c1 = -6;

     out.println(allTheSame(a, b, c));
     out.println(allTheSame(a1, b1, c1));
     out.println(allDifferent(a1, b, c));
     out.println(allDifferent(a, a1, b1));
     out.println(sorted(a, b, c));
     out.println(sorted(a1, a, c1));

     /*
      * Close input and output streams
      */
     in.close();
     out.close();
 }
```