Tracing Example
BallApp/Ball/Baseball/Softball

Program Output

```
Ball constructor 2 19
Bball constructor 10 null
Softball constructor.
Ball constructor 2 20
Ball constructor 2 21
Bball constructor 11 null
Ball constructor 2 22
Ball constructor 2 23
Bball constructor 12 null
Ball constructor 2 24
Bball constructor 13 null
Softball constructor.
Tore cover off!
Hit a mile!
Tore cover off!
Hit a mile!
Tore cover off!
Hit a mile!
class Ball velocity = 3
25
fans = 13 name = null
25
Items = 6
25
fans = 13 name = hard ball
25
13
6
```

Tracing Example
BallApp/Ball/Baseball/Softball

Tracing Steps

BallApp Line 5: **`Softball a = new Softball();`**

- Ball
  - line 4, sets static variable c into memory
- Baseball
  - line 3, sets static variable fans into memory
- Softball
  - line 3, sets static variable items into memory
- BallApp
  - line 5, returns to call to Softball no-arg constructor
- Softball
  - line 7, no-arg Constructor, call to super( ) [for parent Baseball class]
- Baseball
  - line 8, no-arg Constructor, call to super( ) [for parent Ball class]
- Ball
  - line 7, beginning of no-arg Constructor
  - line 3, begins to create object by setting data member velocity into memory with value 2
  - line 8, output statement from no-arg constructor
  - line 11, increment Ball data member velocity
  - line 12, increment Ball static variable c
  - line 13, end of Ball no-arg constructor
- Baseball
  - line 9, Increment Baseball static variable fans
  - line 10, Baseball output statement from within no-arg constructor
  - line 13, end of Baseball no-arg constructor
- Softball
  - line 8, Softball output statement from within no-arg constructor
  - line 10, increment Softball static variable items
  - line 11, end of Softball no-arg constructor
- BallApp
  - line 5, return to BallApp to assign newly created Softball object to Softball reference variable “a”

Resulting Output:

```
Ball constructor 2 19
Bball constructor 10 null
Softball constructor.
```

BallApp Line 6: **`Ball[] b = { new Ball(), new Baseball(), a, new Baseball("hard ball") };`**

Note: in this line an array of type Ball is declared. Each index will serve as a reference variable of objects of type Ball and subclasses of Ball, being Baseball and Softball.

Note: the array stores the addresses of objects, and not the objects themselves.

Ball
- line 7, beginning of no-arg Constructor
- line 3, begins to create object by setting data member velocity into memory with value 2
- line 8, output statement from no-arg constructor
- line 11, increment Ball data member velocity
- line 12, increment Ball static variable c
- line 13, end of Ball no-arg constructor

BallApp
- Line 6, return to continue creating the objects to be stored within the array

Baseball
- line 8, no-arg Constructor, call to super( ) [for parent Ball class]

Ball
- line 7, beginning of no-arg Constructor
- line 3, begins to create object by setting data member velocity into memory with value 2
- line 8, output statement from no-arg constructor
- line 11, increment Ball data member velocity
- line 12, increment Ball static variable c
- line 13, end of Ball no-arg constructor

Baseball
- line 9, Increment Baseball static variable fans
- line 10, Baseball output statement from within no-arg constructor
- line 13, end of Baseball no-arg constructor

BallApp
- line 6
  - return to continue creating the objects to be stored within the array
  - assign address of object created on line 5 as third element of the Ball array

Baseball
- line 16, single-arg Constructor

Ball
- line 7, beginning of no-arg Constructor
- line 3, begins to create object by setting data member velocity into memory with value 2
- line 8, output statement from no-arg constructor
- line 11, increment Ball data member velocity
- line 12, increment Ball static variable c
- line 13, end of Ball no-arg constructor

Baseball
- line 17, assign parameter value to object data member name
- line 18, end of Baseball single-arg constructor

BallApp
- line 6, return to assign the address of the array to the array reference variable b

Resulting Output:

```
Ball constructor 2 20
Ball constructor 2 21
Bball constructor 11 null
Ball constructor 2 22
```

BallApp Line 7: **`Softball b2 = new Softball(5);`**

Softball

line 14, beginning of single-arg constructor

implicit call to parent class [Baseball] no-arg constructor (even though super( ) is not seen in code)

Baseball

line 8, no-arg Constructor, call to super( ) [for parent Ball class]

Ball

line 7, beginning of no-arg Constructor

line 3, begins to create object by setting data member velocity into memory with value 2

line 8, output statement from no-arg constructor

line 11, increment Ball data member velocity

line 12, increment Ball static variable c

line 13, end of Ball no-arg constructor

Baseball

line 9, Increment Baseball static variable fans

line 10, Baseball output statement from within no-arg constructor

line 13, end of Baseball no-arg constructor

Softball

line 15, assign parameter value to static variable items.

Note: class name [Softball] is used to identify items as a static member of the class and not a data member of the current object

Line 16, end of single-arg constructor

BallApp

line 7, return to BallApp to assign newly created Softball object to Softball reference variable “b2”

Resulting Output:

```
Ball constructor 2 23
Bball constructor 12 null
```

BallApp line 8: **`Softball[] sb = { new Softball(), b2 };`**

Note: in this line an array of type Softball is declared. Each index will serve as a reference variable of objects of type Softball.
Note: the array stores the addresses of objects, and not the objects themselves.

Softball
- line 7, no-arg Constructor, call to super( ) [for parent Baseball class]

Baseball
- line 8, no-arg Constructor, call to super( ) [for parent Ball class]

Ball
- line 7, beginning of no-arg Constructor
- line 3, begins to create object by setting data member velocity into memory with value 2
- line 8, output statement from no-arg constructor
- line 11, increment Ball data member velocity
- line 12, increment Ball static variable c
- line 13, end of Ball no-arg constructor

Baseball
- line 9, increment Baseball static variable fans
- line 10, Baseball output statement from within no-arg constructor
- line 13, end of Baseball no-arg constructor

Softball
- line 8, Softball output statement from within no-arg constructor
- line 10, increment Softball static variable items
- line 11, end of Softball no-arg constructor

BallApp
- line 8, return to BallApp to assign newly created Softball object to Softball reference variable “sb[0]” and address of “b2” to “sb[1]”

Resulting Output:
```
Ball constructor 2 24
Bball constructor 13 null
Softball constructor.
```

BallApp Line 10: **`b[1].hit();`**

Baseball

line 22, output statement from within hit( ) method

line 23, call to parent class [Ball] hit( ) method

Ball

line 26, output statement from within hit( ) method

line 27, leaving [Ball] hit( ) method

Baseball

line 24, leaving [Baseball] hit( ) method

Resulting Output:

```
Tore cover off!
Hit a mile!
```

BallApp Line 11: **`sb[0].hit();`**

Baseball

line 22, output statement from within hit( ) method

line 23, call to parent class [Ball] hit( ) method

Ball

line 26, output statement from within hit( ) method

line 27, leaving [Ball] hit( ) method

Baseball

line 24, leaving [Baseball] hit( ) method

Resulting Output:

```
Tore cover off!
Hit a mile!
```

BallApp Line 12: **`((Baseball) sb[1]).hit();`**

- Baseball
  - line 22, output statement from within hit( ) method
  - line 23, call to parent class [Ball] hit( ) method
- Ball
  - line 26, output statement from within hit( ) method
  - line 27, leaving [Ball] hit( ) method
- Baseball
  - line 24, leaving [Baseball] hit( ) method

Note: a double pair of parenthesis is used for casting. The innermost pair around "Baseball" to indicate that "sb[1]" is a reference variable pointing to an object of type [Baseball] (remember that the array type is that of the parent class [Ball]). The outermost pair of parenthesis is around the cast and the array name. This pair is required to instruct the compiler to use the method hit( ) found in class [Baseball] rather than the parent class [Ball].

Resulting Output:

```
Tore cover off!
Hit a mile!
```

BallApp Line 14: **`for (int i = 0; i < b.length; i++)`**

[LOOP ITERATION 1]
BallApp
    line 15, call to class [Ball] toString( ) method
Ball
    line 37, return string from [Ball] toString( ) method
BallApp
    return to line 16 to display string returned from [Ball] line 37 (above)
    line 17, call to class [Ball] getVelocity( ) method
Ball
    line 31, return static class member “c” from class [Ball] getVelocity( ) method
BallApp
    return to line 17 to display value returned from [Ball] line 31 (above)

[LOOP ITERATION 2] [BallApp Line 14]
BallApp
    line 15, call to class [Baseball] toString( ) method
Baseball
    line 29, return string from [Ball] toString( ) method
BallApp
    return to line 16 to display string returned from [Baseball] line 29 (above)
    line 17, call to class [Ball] getVelocity( ) method
Ball
    line 31, return static class member “c” from class [Ball] getVelocity( ) method
BallApp
    return to line 17 to display value returned from [Ball] line 31 (above)

[LOOP ITERATION 3] [BallApp Line 14]
BallApp
    line 15, call to class [Softball] toString( ) method
Softball
    line 21, return string from [Softball] toString( ) method
BallApp
    return to line 16 to display string returned from [Softball] line 21 (above)
    line 17, call to class [Ball] getVelocity( ) method
Ball
    line 31, return static class member “c” from class [Ball] getVelocity( ) method
BallApp
    return to line 17 to display value returned from [Ball] line 31 (above)

[LOOP ITERATION 4] [BallApp Line 14]
BallApp
    line 15, call to class [Baseball] toString( ) method
Baseball
    line 29, return string from [Ball] toString( ) method
BallApp
    return to line 16 to display string returned from [Baseball] line 29 (above)
    line 17, call to class [Ball] getVelocity( ) method
Ball
    line 31, return static class member “c” from class [Ball] getVelocity( ) method
BallApp
    return to line 17 to display value returned from [Ball] line 31 (above)

[LOOP EVALUATION CONTINUED ON NEXT PAGE]

BallApp Line 14: **`for (int i = 0; i < b.length; i++)`**

[LOOP ITERATION 5] fails as condition is no longer true

Resulting Output:

```
class Ball velocity = 3
25
fans = 13 name = null
25
Items = 6
25
fans = 13 name = hard ball
25
```

BallApp Line 20: **`System.out.println(Baseball.fans);`**

BallApp

line 20, display of current value stored in class [Baseball] static variable fans
Note: since the variable is static, the trace does not go into the class. Rather the variable can be called directly using the class name [Baseball]

Resulting Output:

```
13
```

BallApp Line 21: **`System.out.println(Softball.items);`**

BallApp

line 21, display of current value stored in class [Softball] static variable items

Resulting Output:

```
6
```

[END OF PROGRAM]