

CSCI 2320 Java Commands

JavaDoc Comments

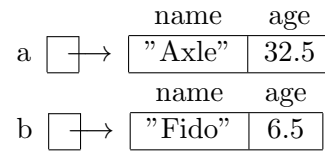
Expected output:

3.1

```
1  /**
2   * Short program to demonstrate JavaDoc comments.
3   *
4   * @author Amy Smith
5   * @version 02 Feb 2020
6   *
7   * <p>Longer descriptions go at bottom inside HTML-style
8   * paragraph tags.</p>
9   */
10
11 public class Demo {
12
13     /**
14      * Calls calcSum method and displays returned result.
15      *
16      * @param args unused
17      */
18     public static void main(String[] args) {
19         System.out.println(calcSum(1.0,2.1));
20     }
21
22     /**
23      * Given to double values returns their sum.
24      *
25      * @param a first value to be used in sum
26      * @param b second value to be used in sum
27      * @return sum of the two parameters
28      */
29     public static double calcSum(double a, double b) {
30         return a+b;
31     }
32 }
```

Classes as Objects (non-OOP)

```
1 public class Animal {
2     String name;
3     double age;
4 }
5
6 public class Driver {
7     public static void main(String[] args) {
8         Animal a,b;
9         a= new Animal();
10        a.name= "Axle";
11        a.age= 32.5;
12        System.out.println(a);
13        System.out.println(a.name);
14        System.out.println(a.age);
15
16        b= new Animal();
17        b.name= "Fido";
18        b.age= 6.5;
19        System.out.println(a.age+b.age);
20    }
21 }
```

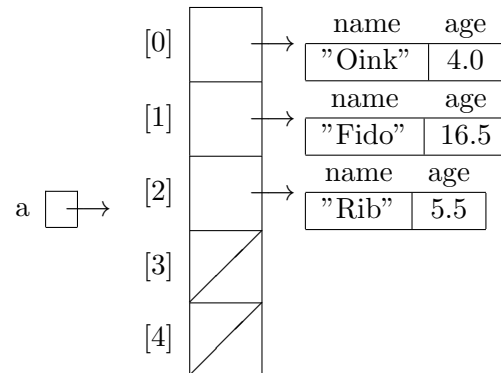


Expected output:

```
Animal@423252d6
Axle
32.5
39.0
```

Arrays of Objects (non-OOP)

```
1 public class Animal {
2     String name;
3     double age;
4 }
5
6 public class Driver {
7     public static void main(String[] args) {
8         int i;
9         Animal [] a;
10        a= new Animal[5];
11
12        a[0]= new Animal();
13        a[0].name= "Oink";
14        a[0].age= 4.0;
15        a[1]= new Animal();
16        a[1].name= "Fido";
17        a[1].age= 16.5;
18        a[2]= new Animal();
19        a[2].name= "Rib";
20        a[2].age= 5.5;
21
22        for (i=0; i<3; i++) {
23            System.out.printf("%-8s %4.1f\n",
24                a[i].name,a[i].age);
25        }
26    }
27 }
```



Expected output:

```
Oink      4.0
Fido     16.5
Rib       5.5
```

Object-Oriented Programming (OOP)

Expected output:

Fido

```
1 public class Animal {
2     private String name;
3     private double age;
4
5     public void setName(String newName) {
6         name= newName;
7     }
8
9     public String getName() {
10        return name;
11    }
12 }
13
14 public class Driver {
15     public static void main(String[] args) {
16         Animal a;
17         a= new Animal();
18         //a.name= "Fido"; // won't compile b/c name is private
19         a.setName("Fido");
20         System.out.println(a.getName());
21     }
22 }
```

Constructors

Expected output:

Not sure is 0.0 years old
Fido is 8.5 years old

```
1 public class Animal {
2     private String name;
3     private double age;
4
5     public Animal() {
6         name= "Not sure";
7         age= 0.0;
8     }
9
10    public Animal(String name, double howOld) {
11        this.name= name;
12        age= howOld;
13    }
14
15    public void show() {
16        System.out.println(name+" is "+age+" years old");
17    }
18 }
19
20 public class Driver {
21     public static void main(String[] args) {
22         Animal a,b;
23         a= new Animal();
24         b= new Animal("Fido",8.5);
25         a.show();
26         b.show();
27     }
28 }
```

Expected output:

Fido is 8.5 years old

```
1 public class Animal {
2     private String name;
3     private double age;
4
5     public Animal(String name, double age) {
6         this.name= name;
7         this.age= age;
8     }
9
10    @Override
11    public String toString() {
12        return name+" is "+age+" years old";
13    }
14 }
15
16 public class Driver {
17     public static void main(String[] args) {
18         Animal a;
19         a= new Animal("Fido",8.5);
20         System.out.println(a);
21     }
22 }
```

```
1 public class Animal {
2     private String name;
3     private double age;
4
5     public Animal(String name, double age) {
6         this.name= name;
7         this.age= age;
8     }
9
10    public double calcEstimatedLifeSpan() {
11        return age*2.0;
12    }
13
14    public double calcEstimatedHeartRate() {
15        return -20.0*calcEstimatedLifeSpan()+460;
16    }
17
18    @Override
19    public String toString() {
20        return name+" is "+age+" years old";
21    }
22 }
23
24 public class Pet extends Animal {
25     private String owner;
26
27     public Pet(String name, double age, String owner) {
28         super(name,age);
29         this.owner= owner;
30     }
31
32     @Override
33     public String toString() {
34         return super.toString()+" (Owner: "+owner+")";
35     }
36 }
37
38 public class Driver {
39     public static void main(String[] args) {
40         Animal a,b;
41         a= new Animal("Algernon",2.2);
42         System.out.println(a+" w/ guessed heart rate of "+
43             a.calcEstimatedHeartRate());
44
45         b= new Pet("Fido",8.5,"Mary");
46         System.out.println(b+" w/ guessed heart rate of "+
47             b.calcEstimatedHeartRate());
48     }
49 }
```

Expected output:

Algernon is 2.2 years old w/ guessed heart rate of 372.0

Fido is 8.5 years old (Owner: Mary) w/ guessed heart rate of 120.0

Expected output:

Fido is 8.5 years old (Owner: Mary)
 Woof, woof!
 Tom is 4.0 years old (Owner: Mary)
 Meow

```

1 public abstract class Animal {
2     private String name;
3     private double age;
4     public Animal(String name, double age) {
5         this.name= name;
6         this.age= age;
7     }
8     public abstract void makeSound();
9
10    @Override
11    public String toString() {
12        return name+" is "+age+" years old";
13    }
14 }
15 public abstract class Pet extends Animal {
16     private String owner;
17     public Pet(String name, double age, String owner) {
18         super(name,age);
19         this.owner= owner;
20     }
21     @Override
22     public String toString() {
23         return super.toString()+" (Owner: "+owner+" ";
24     }
25 }
26 public class Dog extends Pet {
27     public Dog(String name, double age, String owner) {
28         super(name,age,owner);
29     }
30     public void makeSound() {
31         System.out.println("Woof, woof!");
32     }
33 }
34 public class Cat extends Pet {
35     public Cat(String name, double age, String owner) {
36         super(name,age,owner);
37     }
38     public void makeSound() {
39         System.out.println("Meow");
40     }
41 }
42 public class Driver {
43     public static void main(String[] args) {
44         Animal a,b;
45         //a= new Animal(); // no can do
46         //a= new Pet();    // no can do
47         a= new Dog("Fido",8.5,"Mary");
48         b= new Cat("Tom",4.0,"Mary");
49         System.out.println(a);
50         a.makeSound();
51         System.out.println(b);
52         b.makeSound();
53     }
54 }

```

Here is a UML representation of these classes.

Animal
–name: String
–age: double
+calcEstimatedLifeSpan(): double



Pet
–owner: String
+getOwner(): String

```

1 public class Animal {
2     private String name;
3     private double age;
4     public Animal(String name, double age) {
5         this.name= name;
6         this.age= age;
7     }
8     public double calcEstimatedLifeSpan() {
9         return age*2.0;
10    }
11    public double calcEstimatedHeartRate() {
12        return -20.0*calcEstimatedLifeSpan()+460;
13    }
14    @Override
15    public String toString() {
16        return name+" is "+age+" years old";
17    }
18 }
19 public class Pet extends Animal {
20     private String owner;
21     public Pet(String name, double age, String owner) {
22         super(name,age);
23         this.owner= owner;
24     }
25     public String getOwner() {
26         return owner;
27     }
28     @Override
29     public String toString() {
30         return super.toString()+" (Owner: "+owner+")";
31     }
32 }

```

Expected output:

```
Owner: Mary
Age  : 8.5
Span : 17.0
Mary
8.5
17.0
```

```
1 public class Animal {
2     private String name;
3     protected String owner;
4     double age;
5     public double lifeSpan;
6
7     public Animal(String name, double age, String owner) {
8         this.name= name;
9         this.age= age;
10        this.owner= owner;
11        lifeSpan= age*2.0;
12    }
13 }
14 public class Pet extends Animal {
15     public Pet(String name, double age, String owner) {
16         super(name,age,owner);
17     }
18
19     public void show() {
20         //System.out.println("Name : "+name); // nope!
21         System.out.println("Owner: "+owner);
22         System.out.println("Age : "+age);
23         System.out.println("Span : "+lifeSpan);
24     }
25 }
26 public class Driver {
27     public static void main(String[] args) {
28         Pet a;
29         a= new Pet("Fido",8.5,"Mary");
30         a.show();
31         //System.out.println(a.name); // nope
32         System.out.println(a.owner);
33         System.out.println(a.age);
34         System.out.println(a.lifeSpan);
35     }
36 }
```

Expected output:

-12

there is greater than hello

-1

Max is less than Fido

```

1 public class Animal implements Comparable<Animal>{
2     private int id;
3     private String name;
4     private double age;
5
6     public Animal(String name, double age, int id) {
7         this.name= name;
8         this.age= age;
9         this.id= id;
10    }
11
12    @Override
13    public int compareTo(Animal other) {
14        if (this.age > other.age) {
15            return 1;
16        }
17        else if (this.age < other.age) {
18            return -1;
19        }
20        return 0;
21        // return this.name.compareTo(other.name);
22        // return this.id - other.id;
23    }
24
25    @Override
26    public String toString() {
27        return name;
28    }
29 }
30 public class Driver {
31     public static void main(String[] args) {
32         int ans;
33         String x,y;
34         x= "hello";
35         y= "there";
36         ans= x.compareTo(y);
37         System.out.println(ans);
38         if (y.compareTo(x) > 0) {
39             System.out.println(y+" is greater than "+x);
40         }
41
42         Animal a,b;
43         a= new Animal("Max",4.0,20);
44         b= new Animal("Fido",8.5,10);
45         ans= a.compareTo(b);
46         System.out.println(ans);
47         if (a.compareTo(b) < 0) {
48             System.out.println(a+ " is less than "+b);
49         }
50     }
51 }

```

Expected output:

Max(4.0/20) equals Max(1.0/30)

```

1  public class Animal {
2      private int id;
3      private String name;
4      private double age;
5
6      public Animal(String name, double age, int id) {
7          this.name= name;
8          this.age= age;
9          this.id= id;
10     }
11
12     @Override
13     public boolean equals(Object other) {
14         if (this == other) {
15             return true;
16         }
17         if (!(other instanceof Animal)) {
18             return false;
19         }
20         Animal a= (Animal) other;
21         return this.name.equals(a.name);
22     }
23
24     @Override
25     public int hashCode() {
26         return name.hashCode();
27     }
28
29     @Override
30     public String toString() {
31         return name+"("+age+"/"+id+")";
32     }
33 }
34
35 public class Driver {
36     public static void main(String[] args) {
37         Animal a,b,c;
38         a= new Animal("Max",4.0,20);
39         b= new Animal("Fido",8.5,10);
40         c= new Animal("Max",1.0,30);
41         if (a.equals(b)) {
42             System.out.println(a+ " equals "+b);
43         }
44         if (a.equals(c)) {
45             System.out.println(a+ " equals "+c);
46         }
47         if (a == c) {
48             System.out.println(a+ " == "+c);
49         }
50     }
51 }

```

```

1 public class Recursion
2 {
3     public static void main(String [] args)
4     {
5         System.out.println("4-factorial is: "+nfact(4));
6
7         System.out.println("Moving 4 discs from L to R");
8         tower("L", "R", "M", 4);
9     }
10
11    public static void tower(String from, String to, String
12        using, int n)
13    {
14        if (n==1)
15            System.out.println("Move from "+from+" to "+to);
16        else {
17            tower(from,using,to,n-1);
18            tower(from,to,using,1);
19            tower(using,to,from,n-1);
20        }
21    }
22
23    public static int nfact(int n) {
24        if (n==0) {
25            return 1;
26        }
27        return n*nfact(n-1);
28    }
29 }

```

Expected output:

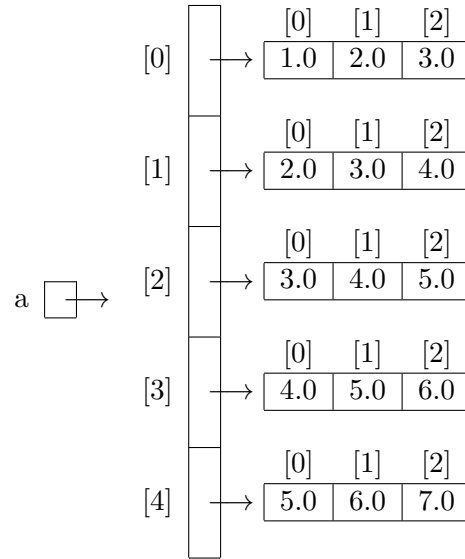
```

4-factorial is: 24
Moving 4 discs from L to R
Move from L to M
Move from L to R
Move from M to R
Move from L to M
Move from R to L
Move from R to M
Move from L to M
Move from L to R
Move from M to R
Move from M to L
Move from R to L
Move from M to R
Move from L to M
Move from L to R
Move from M to R

```

Two-Dimensional Arrays

```
1 public class TwoD
2 {
3     public static void main(String [] args)
4     {
5         int i,j;
6         double [][] a;
7
8         a= new double[5][3];
9
10        for (i=0; i<5; i++) {
11            for (j=0; j<3; j++) {
12                a[i][j]= i+j+1.0;
13            }
14        }
15
16        display(a);
17    }
18
19    public static void display(double [][] a)
20    {
21        int i,j;
22
23        for (i=0; i<a.length; i++)
24        {
25            for (j=0; j<a[i].length; j++) {
26                System.out.printf("%.2f",a[i][j]);
27            }
28            System.out.println();
29        }
30    }
31 }
```



Expected output:

1.00	2.00	3.00
2.00	3.00	4.00
3.00	4.00	5.00
4.00	5.00	6.00
5.00	6.00	7.00

Expected output:

Leon is 8.5 years old
Leon is 2.5 years old
Leon
Leon

```
1 public class Animal {
2     private static String name;
3     private double age;
4
5     public Animal(String name, double age) {
6         this.name= name;
7         this.age= age;
8     }
9
10    public static String getName() {
11        return name;
12    }
13
14    @Override
15    public String toString() {
16        return name+" is "+age+" years old";
17    }
18 }
19
20 public class Driver {
21     public static void main(String[] args) {
22         Animal a,b;
23         a= new Animal("Fido",8.5);
24         b= new Animal("Leon",2.5);
25         System.out.println(a);
26         System.out.println(b);
27         System.out.println(a.getName());
28         System.out.println(Animal.getName());
29     }
30 }
```

Reading Text Files: BufferedReader

Suppose datafile looks like this:

1.5
10.2
2.3
5.2

Expected return value: 19.2

```
1 // requires some imports:
2 import java.io.BufferedReader;
3 import java.io.FileReader;
4 import java.io.IOException;
5
6 public static double sumFile(String filename) throws
7     IOException {
8     BufferedReader numFile;
9     double total=0.0;
10    String str;
11
12    numFile= new BufferedReader(new FileReader(filename));
13    while ((str= numFile.readLine())!=null) {
14        total= total + Double.parseDouble(str);
15    }
16    numFile.close();
17    return total;
18 }
```

Saving Text Files

```
1 // requires some imports:
2 import java.io.FileNotFoundException;
3 import java.io.PrintStream;
4
5 public static void saveData(String filename) throws
    FileNotFoundException
6 {
7     PrintStream f= new PrintStream(filename);
8     double y= 3.8234;
9
10    f.print("This is so fun!");
11    f.print("How are you?\n");
12    f.println("What do we do now?");
13    f.printf("I am am %8.1f feet tall\n",y);
14    f.close();
15 }
```

Expected output (in junk.txt):

This is so fun!How are you?
What do we do now?
I am 3.8 feet tall

Try-Catch

```
1
2 int num;
3 Scanner kb= new Scanner(System.in);
4
5 try {
6     System.out.print("Enter number: ");
7     num= kb.nextInt();
8     System.out.println("You entered: "+num);
9 }
10 catch (InputMismatchException e) {
11     System.out.println("Must enter an integer");
12     System.out.println(e);
13 }
```

Expected output if user enters 7.

Enter number: 7
You entered: 7

Expected output if user enters
"frog".

Enter number: frog
Must be an integer
java.util.InputMismatchException

Linked Lists

```

1 public class ListNode {
2     double data; // this will be a list of double values
3     ListNode next; // reference to the next element
4 }
5
6 ListNode head,p;
7
8 head= new ListNode(); // create first element
9 head.data= 1.5;
10 p= head; // head and both referring to first element
11 p.next= new ListNode(); // create second element
12 p= p.next; // head stays put and p moves to second elem
13 p.data= 2.6;
14 p.next= new ListNode(); // create third element
15 p= p.next; // p moves to third elem
16 p.data= 5.0;
17 p.next= null; // mark this as last element
18
19 // To traverse a list we use a temporary pointer to
20 // start at head and then move through the list to the end.
21 p= head;
22 while (p!=null) {
23     System.out.println(p.data);
24     p= p.next;
25 }

```

Expected output:

1.5
2.6
5.0