CSCI 2320 Java Commands

JavaDoc Comments

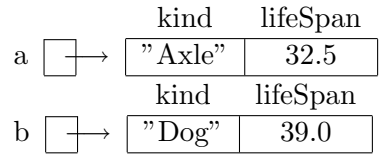
```java
/**
 * Short program to demonstrate JavaDoc comments.
 *
 * @author Amy Smith
 * @version 02 Feb 2020
 *
 * <p>Longer descriptions go at bottom inside HTML-style
 * paragraph tags.</p>
 */

public class Demo {

   /**
    * Calls calcSum method and displays returned result.
    *
    * @param args unused
    */
   public static void main(String[] args) {
      System.out.println(calcSum(1.0,2.1));
   }

   /**
    * Given to double values returns their sum.
    *
    * @param a first value to be used in sum
    * @param b second value to be used in sum
    * @return sum of the two parameters
    */
   public static double calcSum(double a, double b) {
      return a+b;
   }
}
```

Expected output:

```
3.1
```

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 }
```

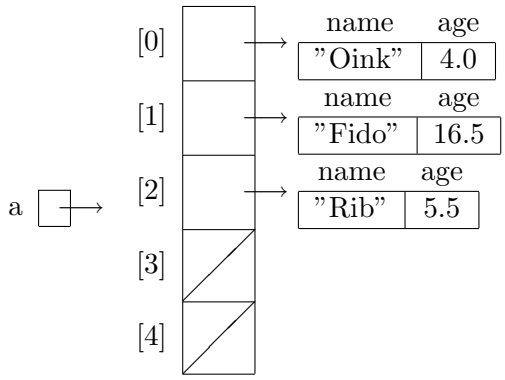


Expected output:

```
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)

```
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 }
```

Expected output:

```
Fido
```

Constructors

```
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:

```
Not sure is 0.0 years old
Fido is 8.5 years old
```

toString()

```
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 }
```

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    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
```

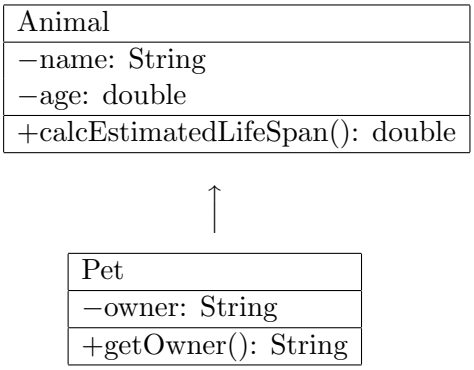
```java
public abstract class Animal {
   private String name;
   private double age;
   public Animal(String name, double age) {
      this.name= name;
      this.age= age;
   }
   public abstract void makeSound();

   @Override
   public String toString() {
      return name+" is "+age+" years old";
   }
}
public abstract class Pet extends Animal {
   private String owner;
   public Pet(String name, double age, String owner) {
      super(name,age);
      this.owner= owner;
   }
   @Override
   public String toString() {
      return super.toString()+" (Owner: "+owner+")";
   }
}
public class Dog extends Pet {
   public Dog(String name, double age, String owner) {
      super(name,age,owner);
   }
   public void makeSound() {
      System.out.println("Woof, woof!");
   }
}
public class Cat extends Pet {
   public Cat(String name, double age, String owner) {
      super(name,age,owner);
   }
   public void makeSound() {
      System.out.println("Meow");
   }
}
public class Driver {
   public static void main(String[] args) {
      Animal a,b;
      //a= new Animal(); // no can do
      //a= new Pet();    // no can do
      a= new Dog("Fido",8.5,"Mary");
      b= new Cat("Tom",4.0,"Mary");
      System.out.println(a);
      a.makeSound();
      System.out.println(b);
      b.makeSound();
   }
}
```

Expected output:

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

```java
public class Animal {
   private String name;
   private double age;
   public Animal(String name, double age) {
      this.name= name;
      this.age= age;
   }
   public double calcEstimatedLifeSpan() {
      return age*2.0;
   }
   public double calcEstimatedHeartRate() {
      return -20.0*calcEstimatedLifeSpan()+460;
   }
   @Override
   public String toString() {
      return name+" is "+age+" years old";
   }
}
public class Pet extends Animal {
   private String owner;
   public Pet(String name, double age, String owner) {
      super(name,age);
      this.owner= owner;
   }
   public String getOwner() {
      return owner;
   }
   @Override
   public String toString() {
      return super.toString()+" (Owner: "+owner+")";
   }
}
```

Here is a UML representation of these classes.



Access Modifiers

```
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:

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

Comparable

```java
public class Animal implements Comparable<Animal>{
   private int id;
   private String name;
   private double age;

   public Animal(String name, double age, int id) {
      this.name= name;
      this.age= age;
      this.id= id;
   }

   @Override
   public int compareTo(Animal other) {
      if (this.age > other.age) {
         return 1;
      }
      else if (this.age < other.age) {
         return -1;
      }
      return 0;
      // return this.name.compareTo(other.name);
      // return this.id - other.id;
   }

   @Override
   public String toString() {
      return name;
   }
}
public class Driver {
   public static void main(String[] args) {
      int ans;
      String x,y;
      x= "hello";
      y= "there";
      ans= x.compareTo(y);
      System.out.println(ans);
      if (y.compareTo(x) > 0) {
         System.out.println(y+" is greater than "+x);
      }

      Animal a,b;
      a= new Animal("Max",4.0,20);
      b= new Animal("Fido",8.5,10);
      ans= a.compareTo(b);
      System.out.println(ans);
      if (a.compareTo(b) < 0) {
         System.out.println(a+ " is less than "+b);
      }
   }
}
```

Expected output:

```
-12
there is greater than hello
-1
Max is less than Fido
```

Equals

```java
public class Animal {
   private int id;
   private String name;
   private double age;

   public Animal(String name, double age, int id) {
      this.name= name;
      this.age= age;
      this.id= id;
   }

   @Override
   public boolean equals(Object other) {
      if (this == other) {
         return true;
      }
      if (!(other instanceof Animal)) {
         return false;
      }
      Animal a= (Animal) other;
      return this.name.equals(a.name);
   }

   @Override
   public int hashCode() {
      return name.hashCode();
   }

   @Override
   public String toString() {
      return name+"("+age+"/"+id+")";
   }
}

public class Driver {
   public static void main(String[] args) {
      Animal a,b,c;
      a= new Animal("Max",4.0,20);
      b= new Animal("Fido",8.5,10);
      c= new Animal("Max",1.0,30);
      if (a.equals(b)) {
         System.out.println(a+ " equals "+b);
      }
      if (a.equals(c)) {
         System.out.println(a+ " equals "+c);
      }
      if (a == c) {
         System.out.println(a+ " == "+c);
      }
   }
}
```

Expected output:

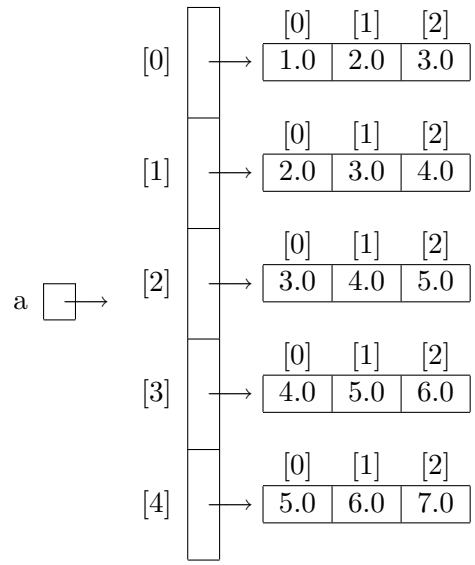
```
Max(4.0/20) equals Max(1.0/30)
```

```java
public class Recursion
{
   public static void main(String [] args)
   {
      System.out.println("4-factorial is: "+nfact(4));

      System.out.println("Moving 4 discs from L to R");
      tower("L","R","M",4);
   }

   public static void tower(String from, String to, String
       using, int n)
   {
      if (n==1)
         System.out.println("Move from "+from+" to "+to);
      else {
         tower(from,using,to,n-1);
         tower(from,to,using,1);
         tower(using,to,from,n-1);
      }
   }

   public static int nfact(int n) {
      if (n==0) {
         return 1;
      }
      return n*nfact(n-1);
   }
}
```

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
```

```
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("%6.2f",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
```

Static

```
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 }
```

Expected output:

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

Reading Text Files: BufferedReader

```
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
       IOException {
7     BufferedReader numFile;
8     double total=0.0;
9     String str;
10
11    numFile= new BufferedReader(new FileReader(filename));
12    while ((str= numFile.readLine())!=null) {
13       total= total + Double.parseDouble(str);
14    }
15    numFile.close();
16    return total;
17 }
```

Suppose datafile looks like this:

```
1.5
10.2
2.3
5.2
```

Expected return value: 19.2

Saving Text Files

```
   // requires some imports:
import java.io.FileNotFoundException;
import java.io.PrintStream;

public static void saveData(String filename) throws
    FileNotFoundException
{
   PrintStream f= new PrintStream(filename);
   double y= 3.8234;

   f.print("This is so fun!");
   f.print("How are you?\n");
   f.println("What do we do now?");
   f.printf("I am am %8.1f feet tall\n",y);
   f.close();
}
```

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

```

int num;
Scanner kb= new Scanner(System.in);

try {
   System.out.print("Enter number: ");
   num= kb.nextInt();
   System.out.println("You entered: "+num);
}
catch (InputMismatchException e) {
   System.out.println("Must enter an integer");
   System.out.println(e);
}
```

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
```