LINKED OBJECTS

CS210 – Data Structures and Algorithms

Dr. Basit Qureshi

https://www.drbasit.org/

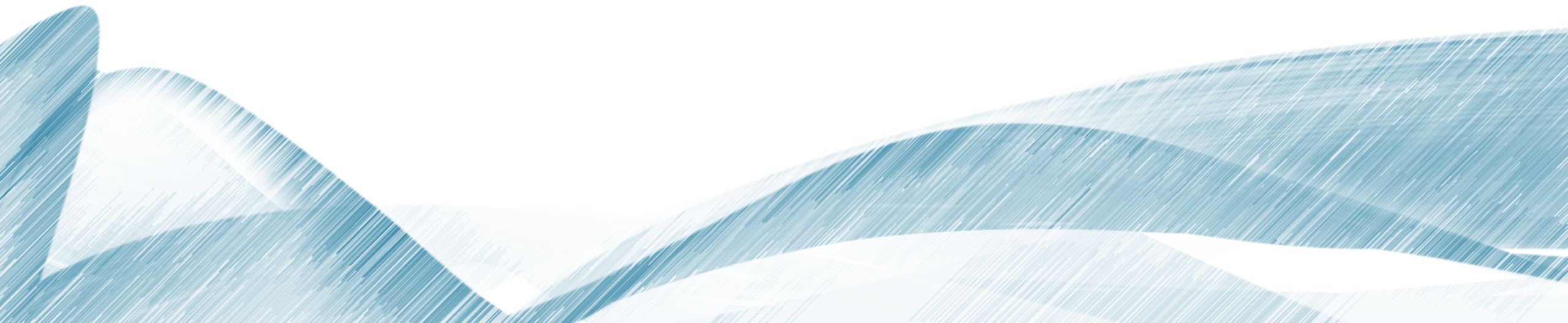
TOPICS

- Java data types
- Memory allocation
  - Primitive Data Type
  - Arrays of Primitive Data Types
  - Object Data Type
  - Arrays of Object Data Types
- Linked objects
- Linked List

JAVA - DATA TYPES

Type	Size in Bytes	Range
`byte`	1 byte	-128 to 127
`short`	2 bytes	-32,768 to 32,767
`int`	4 bytes	-2,147,483,648 to 2,147,483, 647
`long`	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`float`	4 bytes	approximately ±3.40282347E+38F (6-7 significant decimal digits) *Java implements IEEE 754 standard*
`double`	8 bytes	approximately ±1.7976931348623157E+308 (15 significant decimal digits)
`char`	2 byte	0 to 65,536 (unsigned)
`boolean`	not precisely defined*	true or false

MEMORY ALLOCATION

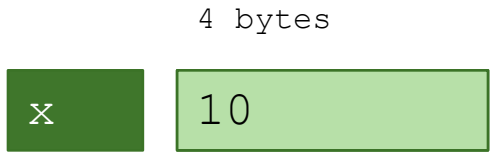


MEMORY ALLOCATION

- Primitive data types

```
int x;

x = 10;
```



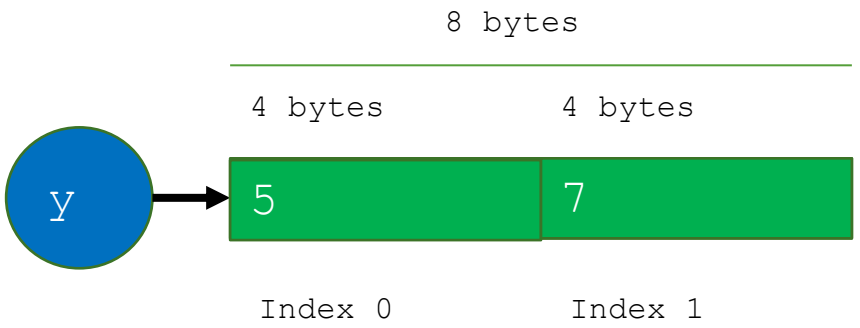
MEMORY ALLOCATION

- Single Dimension array of a Primitive data type

```
int [] y;

y = new int [2];

y = 5; // Error
y[0] = 5;
y[1] = 7;
```



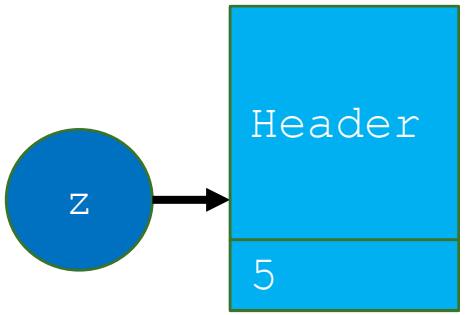
y is a **reference** to **memory location** of the array

MEMORY ALLOCATION

```
public class obj {
   int val;
   public obj(){
     val = 0;
   }
}
```

- Object data types

```
obj z;
z = new obj();
z.val = 5;
```



64-bit JDK defines minimum header size = 16 bytes
* Padding is also used to make the total size divisible by 8.
The actual size depends on 32 or 64 bit VM arch.

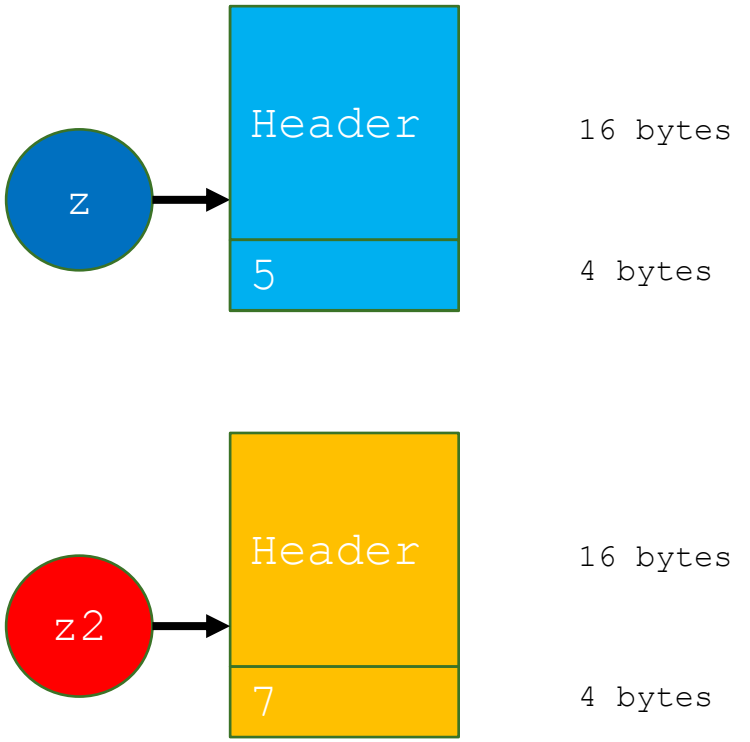
MEMORY ALLOCATION

```
public class obj {
   int val;
   public obj(){
      val = 0;
   }
   public obj(int x){
      val = x;
   }
}
```

- Object data types

```
obj z;
z = new obj();
z.val = 5;

obj z2 = new obj(7);
```

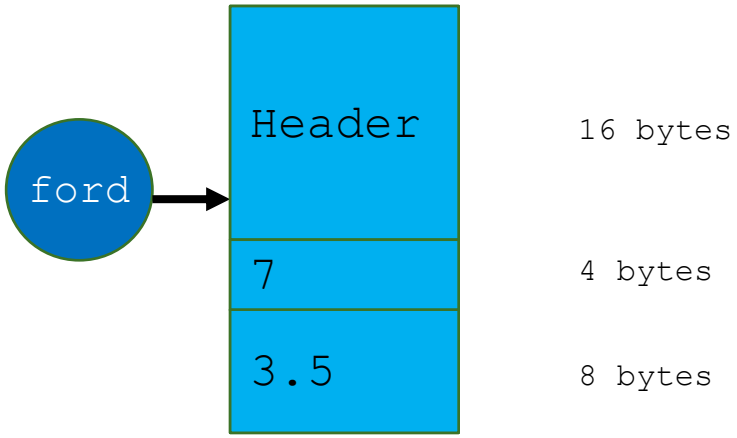


MEMORY ALLOCATION

```
public class car {
   int seats;
   double engine;
   public car(){
      seats= 4;
      engine= 2.0;
   }
   public car(int x,
double e){
      seats= x;
      engine = e;
   }
}
```

- Object data types

```
car ford;
ford= new car(7, 3.5);
```



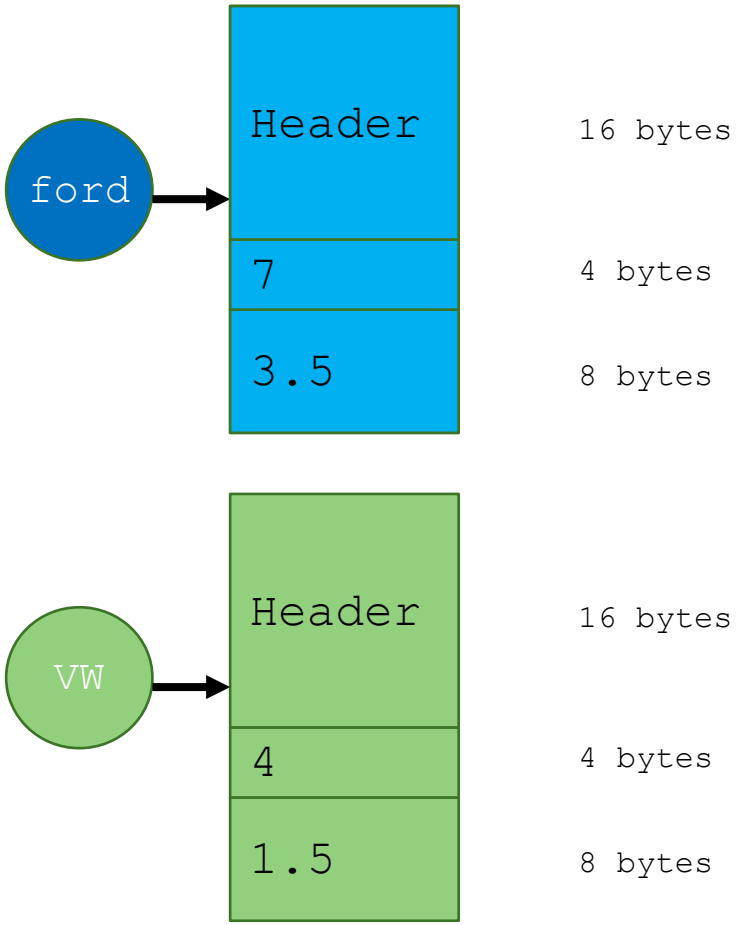
MEMORY ALLOCATION

```
public class car {
   int seats;
   double engine;
   public car(){
      seats= 4;
      engine= 2.0;
   }
   public car(int x,
double e){
      seats= x;
      engine = e;
   }
}
```

- Object data types

```
car ford;
ford= new car(7, 3.5);

car VW = new car(4, 1.5);
```



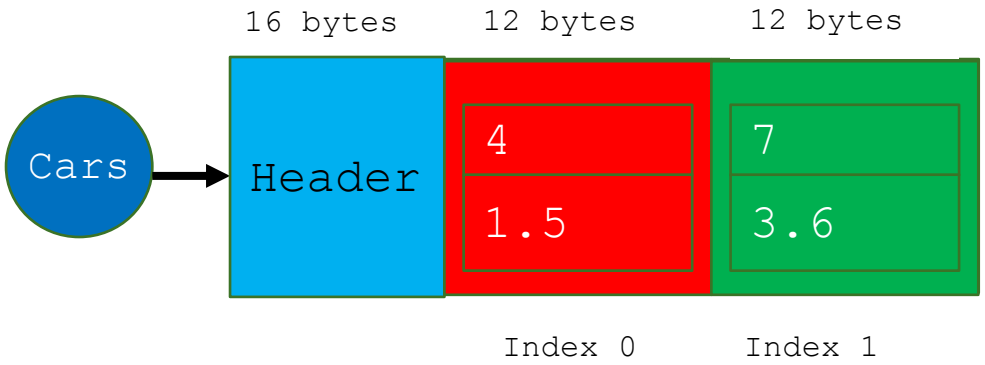
MEMORY ALLOCATION

```
public class car {
   int seats;
   double engine;
   public car(){
      seats= 4;
      engine= 2.0;
   }
   public car(int x,
double e){
      seats= x;
      engine = e;
   }
}
```

- Single dimension array of an object data type

```
car [] Cars;
Cars= new car [2];

Cars[0].seats=4;
Cars[0].engine=1.5;
Cars[1].seats=7;
Cars[1].engine=3.6;
```



EXERCISE

```
public class car {
   int seats;
   double engine;
   public car(){
      seats= 4;
      engine= 2.0;
   }
   public car(int x,
double e){
      seats= x;
      engine = e;
   }
}
```

Exercise:

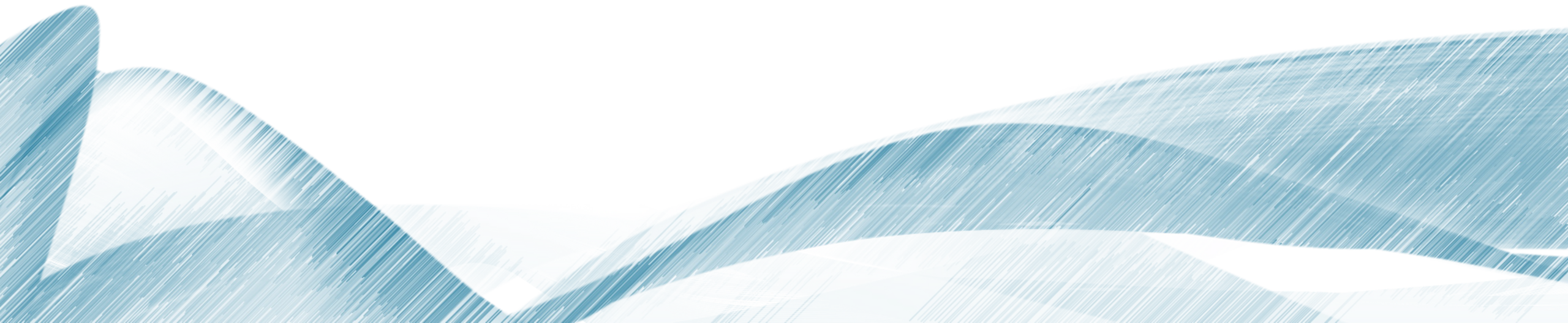
```
car [] Toyota = new Toyota [5];
```

- What is the size of the Toyota object in bytes?

Header	16 bytes
Index 0	4 + 8 =12 bytes
Index 1	4 + 8 =12 bytes
Index 2	4 + 8 =12 bytes
Index 3	4 + 8 =12 bytes
Index 4	4 + 8 =12 bytes

76 bytes

LINKED OBJECTS



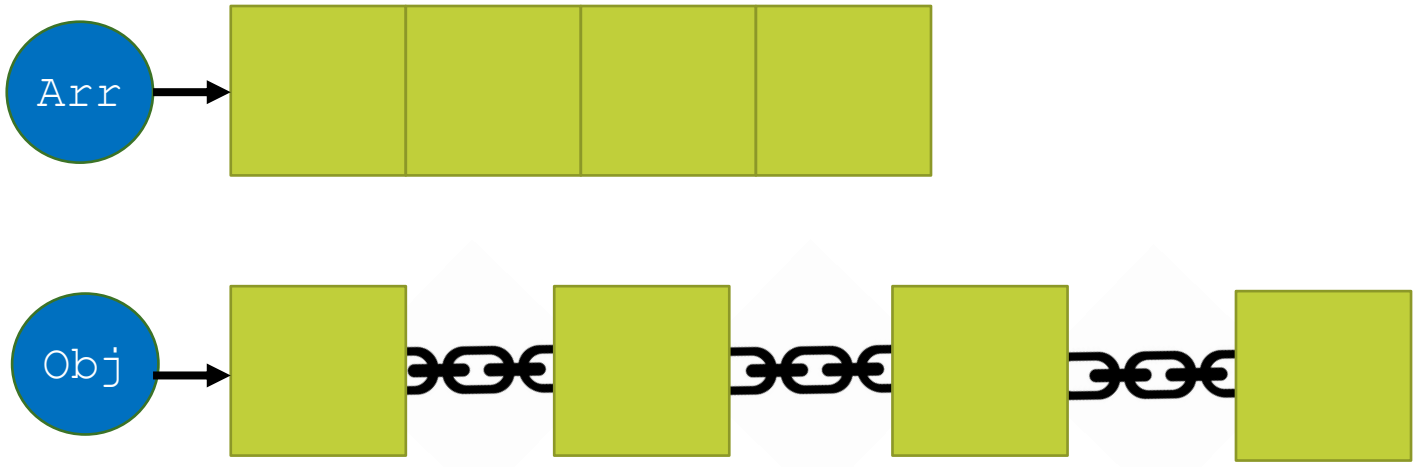
PROBLEM WITH ARRAYS

1. Arrays need to be defined before usage.
2. Fixed size: relocation is expensive

What if… you **do not know** **the size** of **your data** ?

LINKED OBJECTS

- Solution…. Linked Objects

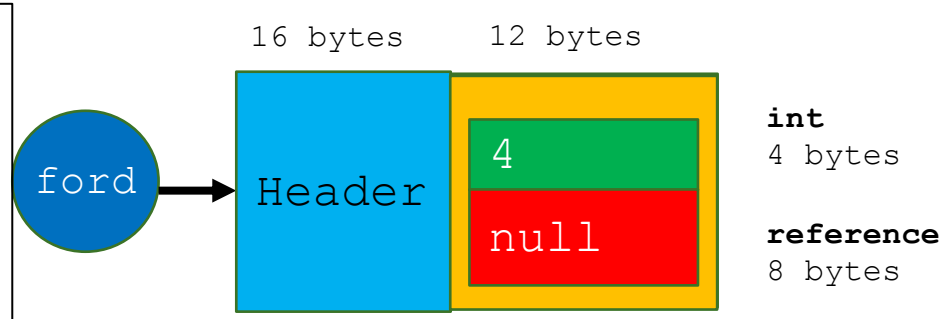


LINKED OBJECTS

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Making a simple linked object

```
car ford = new car();
ford.val = 4;
ford.next = null;
```



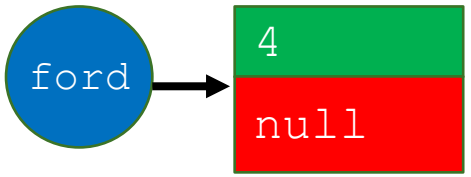
LINKED OBJECTS

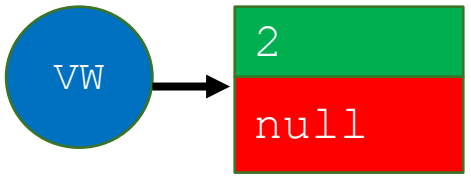
```
public class car {
  int val;
  car next;
  public car(){
    val = 0;
    next = null;
  }
}
```

- Linking two objects

```
car ford = new car();
ford.val = 4;
ford.next = null;

car VW = new car();
VW.val = 2;
VW.next = null;
```





LINKED OBJECTS

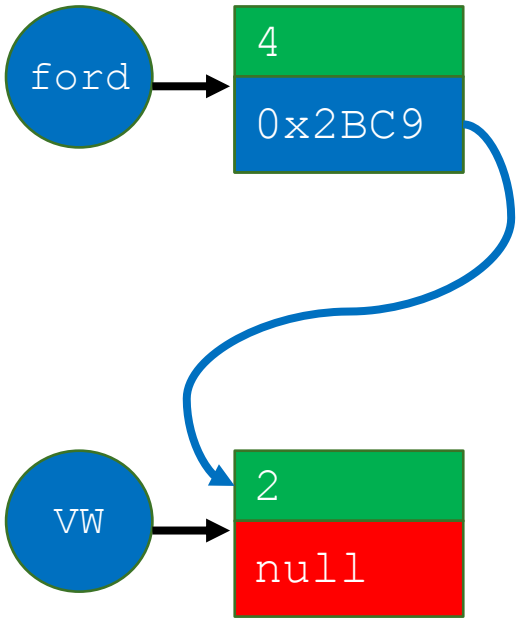
```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Linking two objects

```
car ford = new car();
ford.val = 4;
ford.next = null;

car VW = new car();
VW.val = 2;
VW.next = null;

ford.next = VW;
```



Address used is a hypothetical value

LINKED OBJECTS

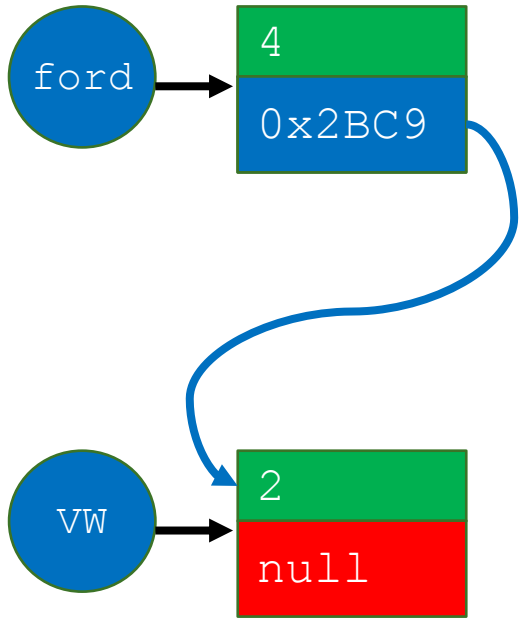
```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Linking two objects

```
car ford = new car();
ford.val = 4;
ford.next = null;

car VW = new car();
VW.val = 2;
VW.next = null;

ford.next = VW;
```



TRIVIA

ford.next is VW
(ford.next).val is VW.val
So
ford.next.val is 2

LINKED OBJECTS

```
public class car {
  int val;
  car next;
  public car(){
    val = 0;
    next = null;
  }
}
```

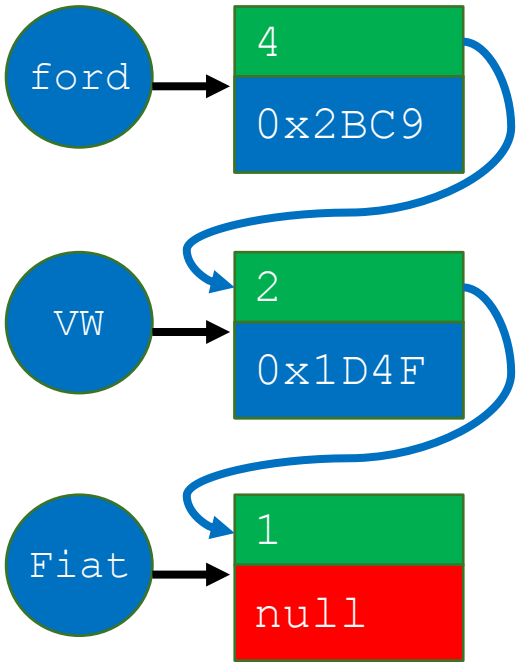
- Linking three objects

```
car ford = new car();
ford.val = 4; ford.next = null;

car VW = new car();
VW.val = 2; VW.next = null;

car Fiat = new car();
Fiat.val = 1; Fiat.next = null;

ford.next = VW;
VW.next = Fiat;
```



LINKED OBJECTS

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

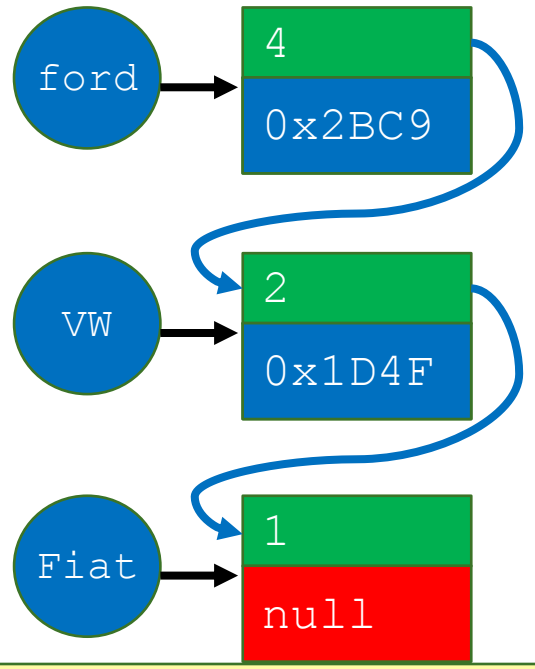
- Linking three objects

```
car ford = new car();
ford.val = 4; ford.next = null;

car VW = new car();
VW.val = 2; VW.next = null;

car Fiat = new car();
Fiat.val = 1; Fiat.next = null;

ford.next = VW;
VW.next = Fiat;
```



TRIVIA

`ford.next.val` is `VW.val`
`ford.next.next.val` is `Fiat.val`

EXERCISE

```
public class car {
  int val;
  car next;
  public car(){
    val = 0;
    next = null;
  }
}
```

- Exercise

Make four objects of type `car` and link them…

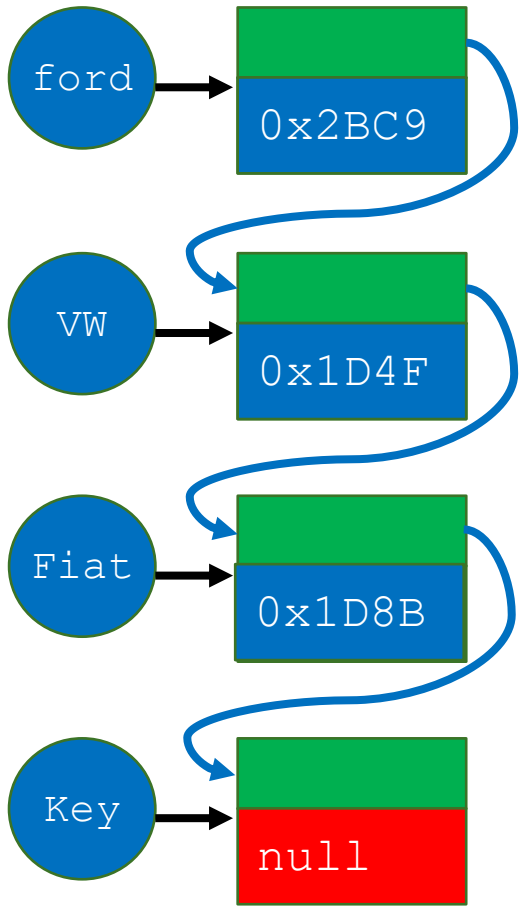
LINKED OBJECTS

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

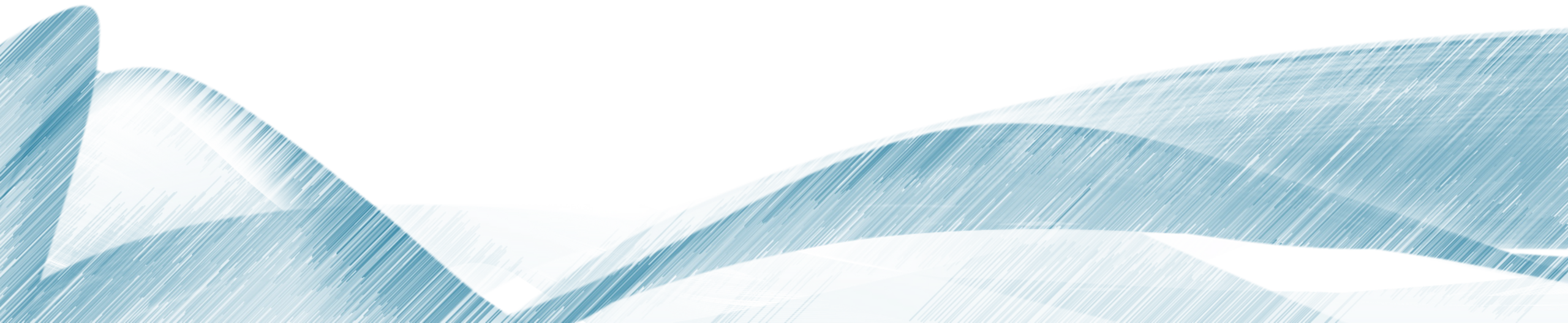
- Exercise

Make four objects of type `car` and link them…

```
car ford = new car();
car VW = new car();
car Fiat = new car();
car Key = new car();
ford.next = VW;
VW.next = Fiat;
Fiat.next = Key;
```



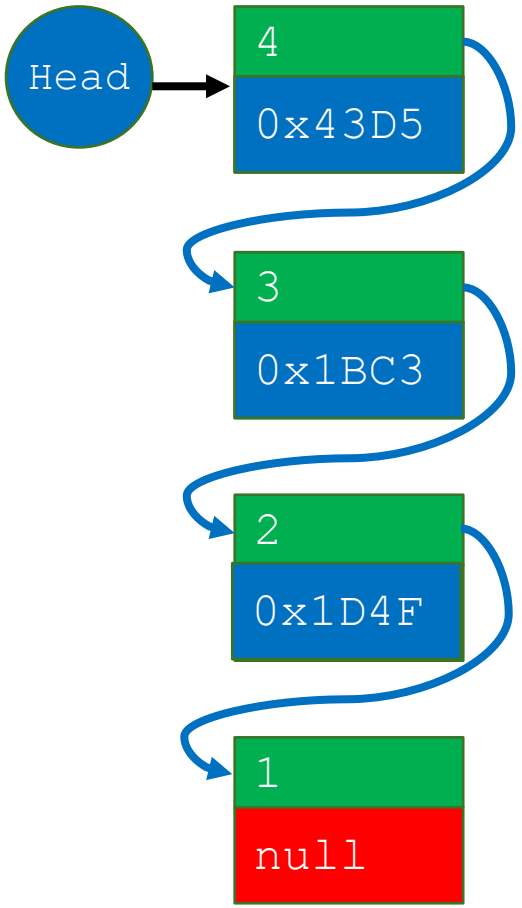
LINKED LIST



LINKED LIST

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- A Linked List connects many linked objects using a Head.



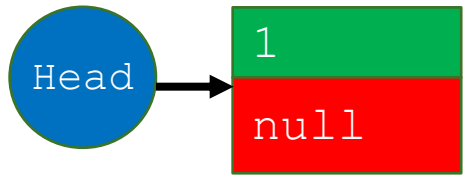
LINKED LIST

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Linked List

Stores linked nodes using a Head

```
car Head = new car();
Head.val = 1; Head.next = null;
```



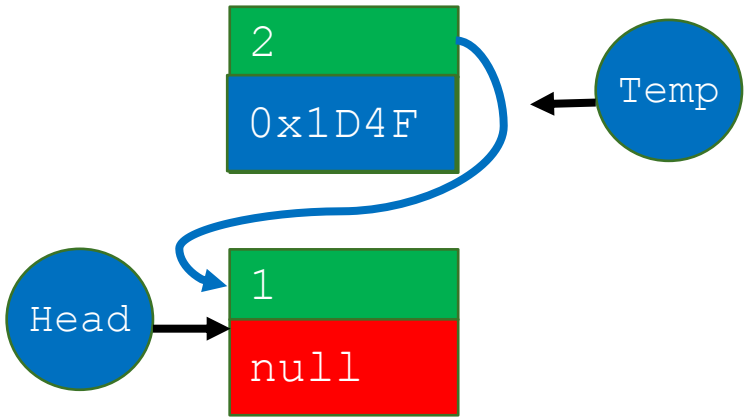
LINKED LIST

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Linked List

Stores linked nodes using a Head

```
car Head = new car();
Head.val = 1; Head.next = null;
car Temp = new car();
Temp.val = 2; Temp.next = Head;
Head = Temp;
```



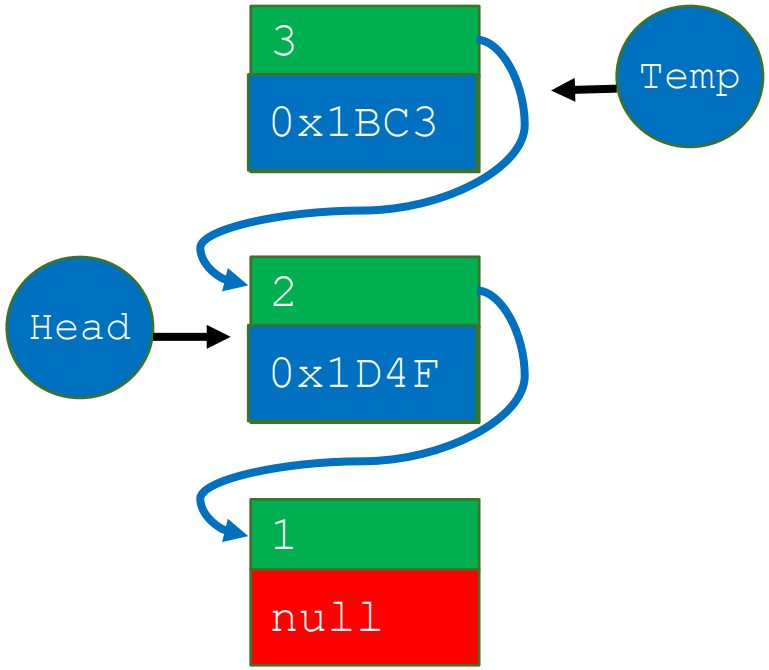
LINKED LIST

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Linked List

Stores linked nodes using a Head

```
car Head = new car();
Head.val = 1; Head.next = null;
car Temp = new car();
Temp.val = 2; Temp.next = Head;
Head = Temp;
car Temp = new car();
Temp.val = 3; Temp.next = Head;
Head = Temp;
```



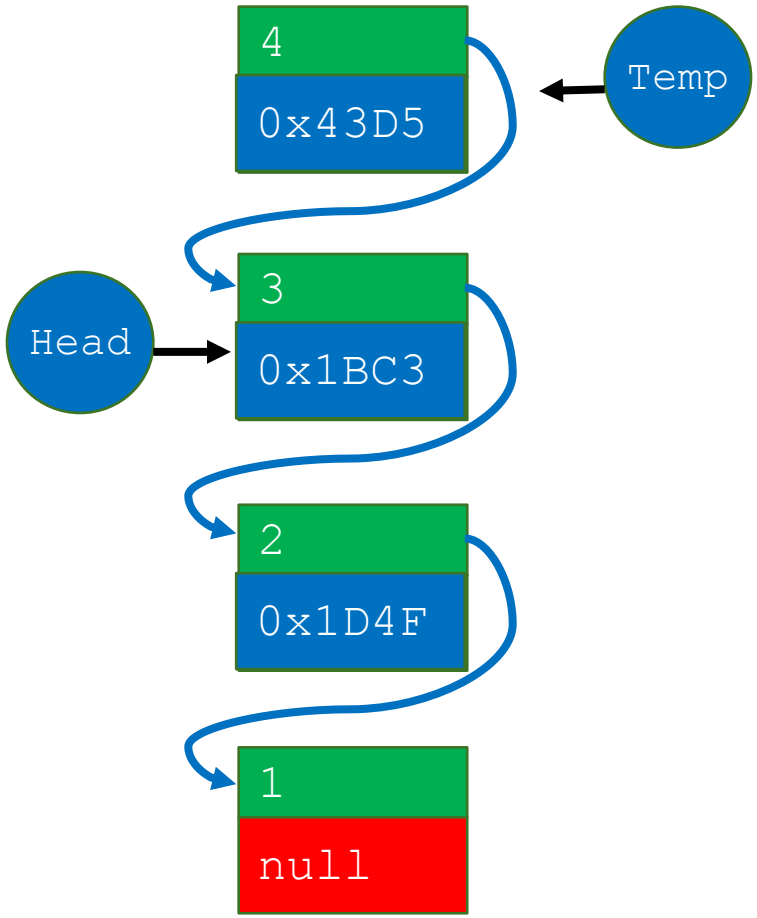
LINKED LIST

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```

- Linked List

Stores linked nodes using a Head

```
car Head = new car();
Head.val = 1; Head.next = null;
car Temp = new car();
Temp.val = 2; Temp.next = Head;
Head = Temp;
car Temp = new car();
Temp.val = 3; Temp.next = Head;
Head = Temp;
car Temp = new car();
Temp.val = 4; Temp.next = Head;
Head = Temp;
```



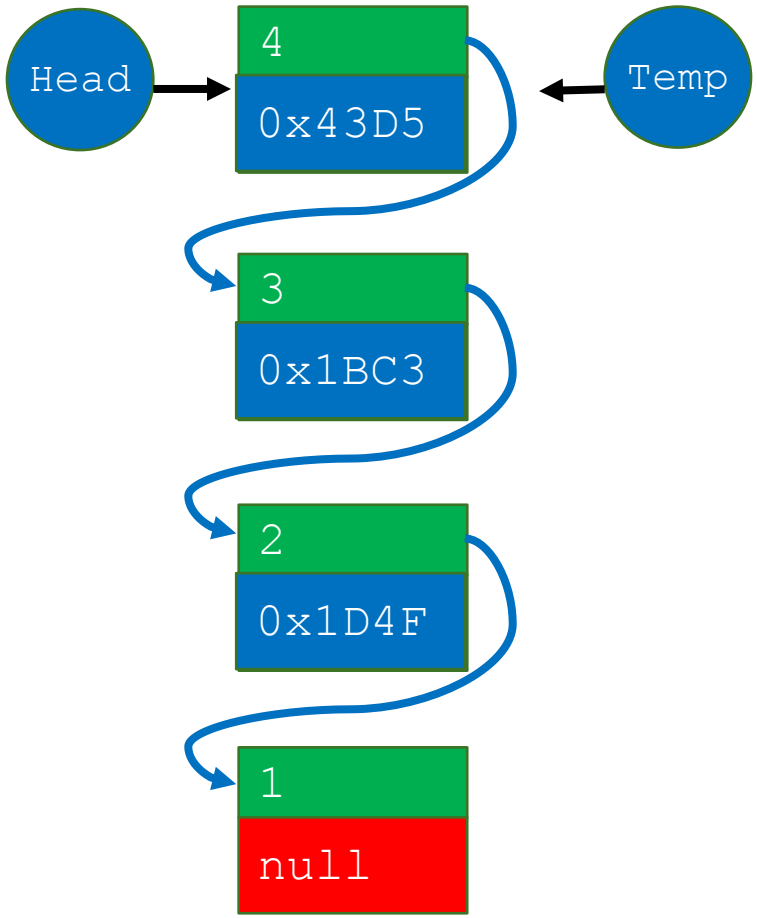
LINKED LIST

```
public class car {
   int val;
   car next;
   public car(){
      val = 0;
      next = null;
   }
}
```


- Linked List

Stores linked nodes using a Head

```
car Head = new car();
Head.val = 1; Head.next = null;
car Temp = new car();
Temp.val = 2; Temp.next = Head;
Head = Temp;
car Temp = new car();
Temp.val = 3; Temp.next = Head;
Head = Temp;
car Temp = new car();
Temp.val = 4; Temp.next = Head;
Head = Temp;
```

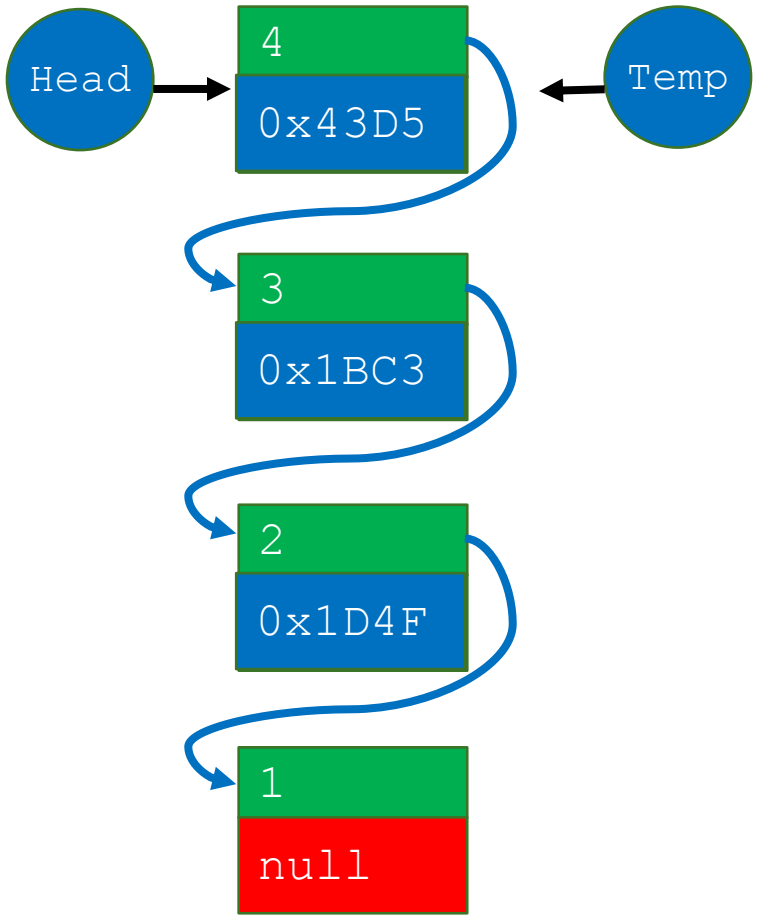


LINKED LIST

```
public class car {
  int val;
  car next;
  public car(){
    val = 0;
    next = null;
  }
 public car(int x, car nxt){
    val = x;
    next = nxt;
  }
}
```

- Linked List: A simplified look!

```
car Head = new car(1,null);
car Temp;
for(int n = 2; n<=4; n++){
  Temp = new car(n, Head);
  Head = Temp;
}
```

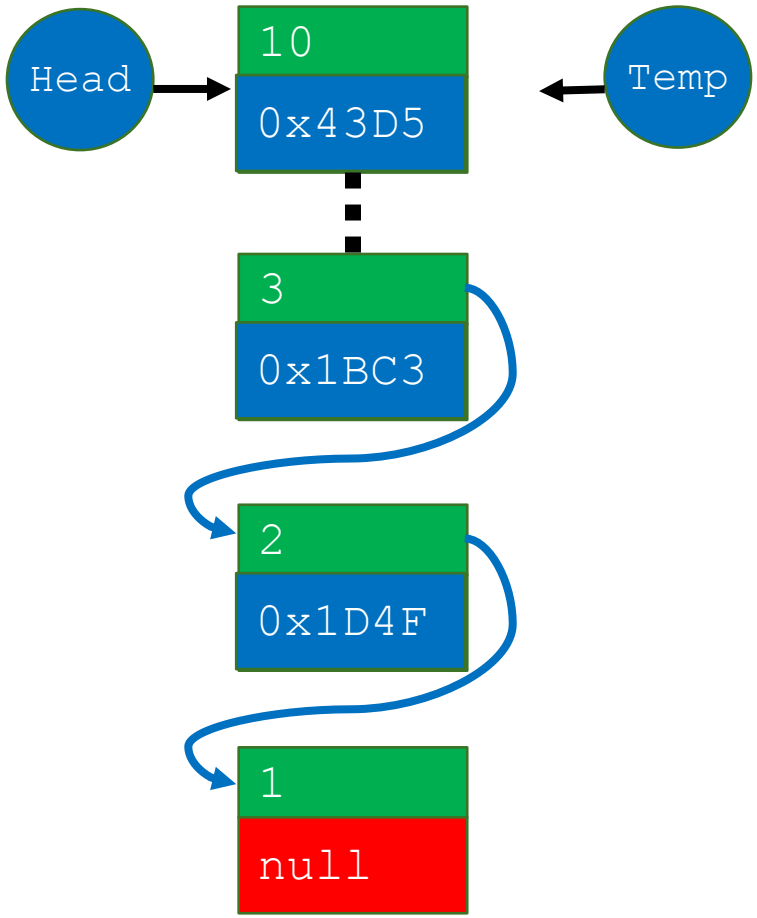


EXERCISE

```
public class car {
  int val;
  car next;
  public car(){
    val = 0;
    next = null;
  }
 public car(int x, car
nxt){
    val = x;
    next = nxt;
  }
}
```

- Make a List of 10 connected objects storing values 1 to 10.

```
int size = 10;
car Head = new car(1,null);
for(int n = 2; n<=size; n++){
  car Temp = new car(n, Head);
  Head = Temp;
}
```



The End!