UNIVERSITY OF Hull

Objects and References

Rob Miles

Department of Computer Science

28b 08120 Programming 2

Objects and References

- Objects let us lump data together into values that contain multiple fields
- Variables can then contain values which are made up of a set of related items
- Now we are going to find out how to use references to allow us to handle very large amounts of data

The Account Class

```
class Account
{
    public string Name;
    public int AccountNumber;
    public int Balance;
}
```

- We have seen structures before
  - They allow us to create “lumps of data”
- If all we need to store is the name of the account holder, their account number and the amount of money they have we can create a structure like the one above

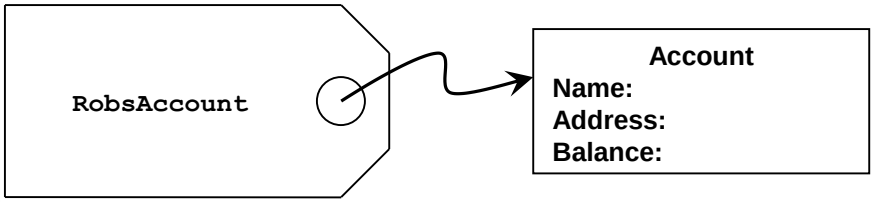
Creating an Account Reference

```
Account RobsAccount;
```

- Declaring a variable of type `Account` creates a tag that can refer to `Account` instances
- It does **not** create anything that can store account information

Creating an Account Instance

```
Account RobsAccount;
RobsAccount = new Account();
```

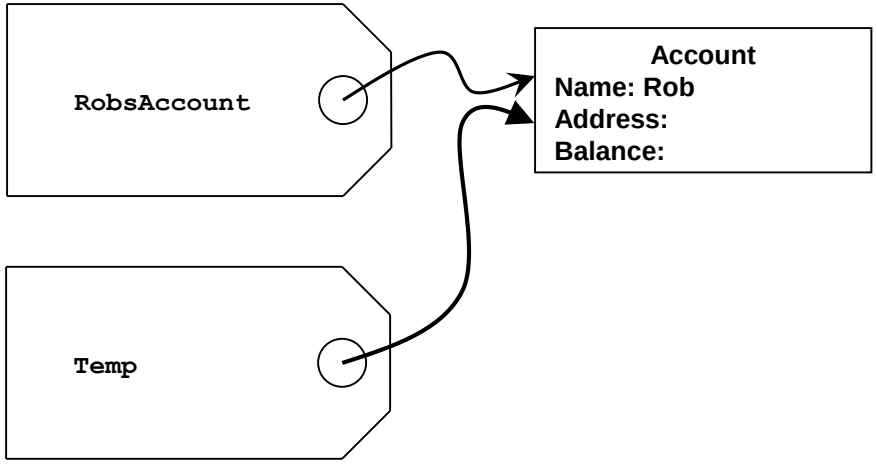


- If we want an `Account` instance we have to create it and set the reference to refer to it
- Note that this is different from structures (and other types managed by value)

Using Duplicate References

```
Account RobsAccount;
RobsAccount = new Account();

Account Temp;
Temp = RobsAccount;
Temp.Name = "Rob";
```

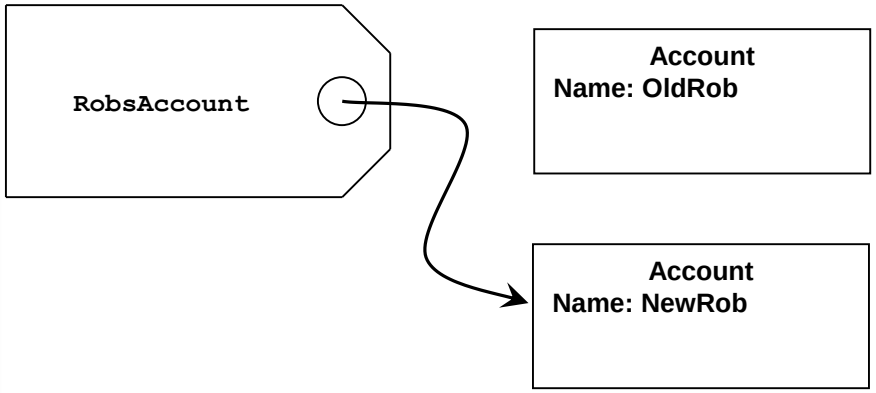


- An object in memory can have multiple references referring to it
- In the above code we have changed the `Name` property of the object referred to by `Temp`
- We could have changed the name of `RobsAccount`

Removing References

```
Account RobsAccount;
RobsAccount = new Account();
RobsAccount.Name = "OldRob";

RobsAccount = new Account();
RobsAccount.Name = "NewRob";
```



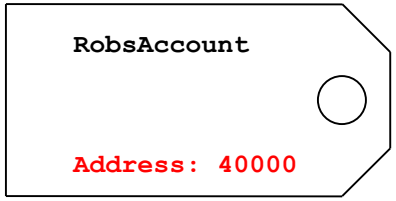
- The above code creates two objects
- When the code has completed only one of the objects has a reference to it
- The Account with the name OldRob is no longer accessible
- It will be removed automatically by the Garbage Collector process

Garbage Collection

- The Garbage Collector is a process that runs alongside your program
- It constantly looks for objects that no longer have references
- These are automatically removed from memory
- Not all languages have automatic garbage collection
  - C and C++ do not provide this
- In those languages your program must explicitly dispose of objects that are no longer required

How References Really Work

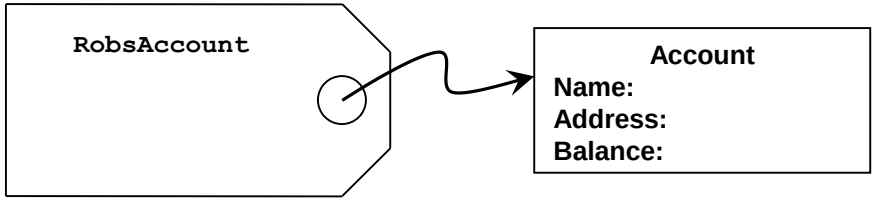
```
Account RobsAccount;
```



- We can view a reference as a Tag, and the connection between the reference and the object as a “rope” that is tied from the tag to the object

How References Really Work

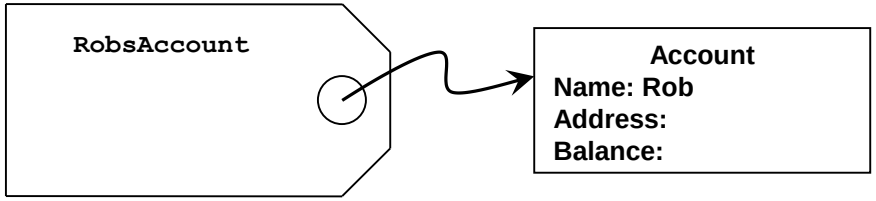
```
Account RobsAccount;
RobsAccount = new Account();
```



- We can view a reference as a Tag, and the connection between the reference and the object as a “rope” that is tied from the tag to the object
- When you assign a reference it is like tying the tag to the object

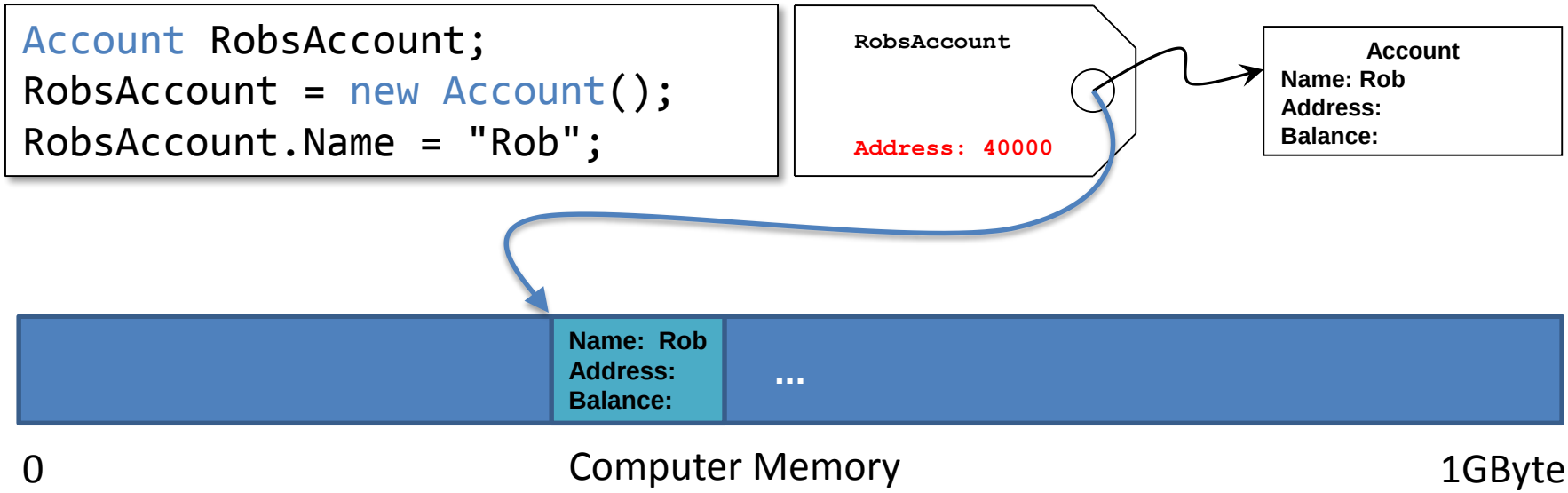
How References Really Work

```
Account RobsAccount;
RobsAccount = new Account();
RobsAccount.Name = "Rob";
```



- We can view a reference as a Tag, and the connection between the reference and the object as a “rope” that is tied from the tag to the object
- When you assign a reference it is like tying the tag to the object
- When you access a property via a reference the reference is used to find the object that is being used

How References Really Work



- References actually work by holding the address in memory of the object
- However, the physical location of your object is hidden from your program

Why Bother with References?

- References seem to make life more difficult:
  - We have to create the objects before we can use them
  - We can confuse ourselves by having more than one reference to a single object
  - We can inadvertently let go of a reference and lose the item on the end of it
  - The Garbage Collector has to come along and remove unused objects
- However, references are actually very useful when it comes to managing large amounts of data

Creating a Bank with Structures

```
AccountStruct [] Bank = new AccountStruct [100];
```

- If we want to store 100 bank accounts we need an array to hold 100 of them
- If we use structures this is very easy, we just have to create an array of the appropriate size
- Because structures are managed by value this will create the required number of accounts

Storing Structures in Memory

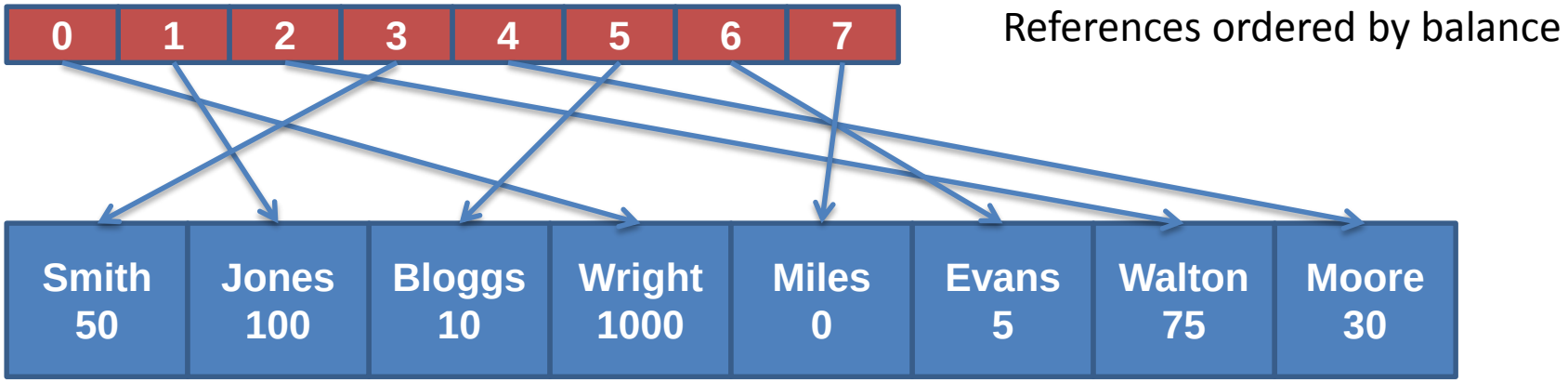
- When we create an array of structures they are stored in a single block of memory
- The array subscript is used to identify the particular part of the block that holds that element
- Above shows the storage that would be used to hold 100 account values
- Note that they are numbered from 0

Sorting Stored Structures

Bloggs 10	Evans 5	Jones 100	Miles 0	Moore 30	Smith 50	Walton 75	Wright 1000
0	1	2	3	4	5	6	7

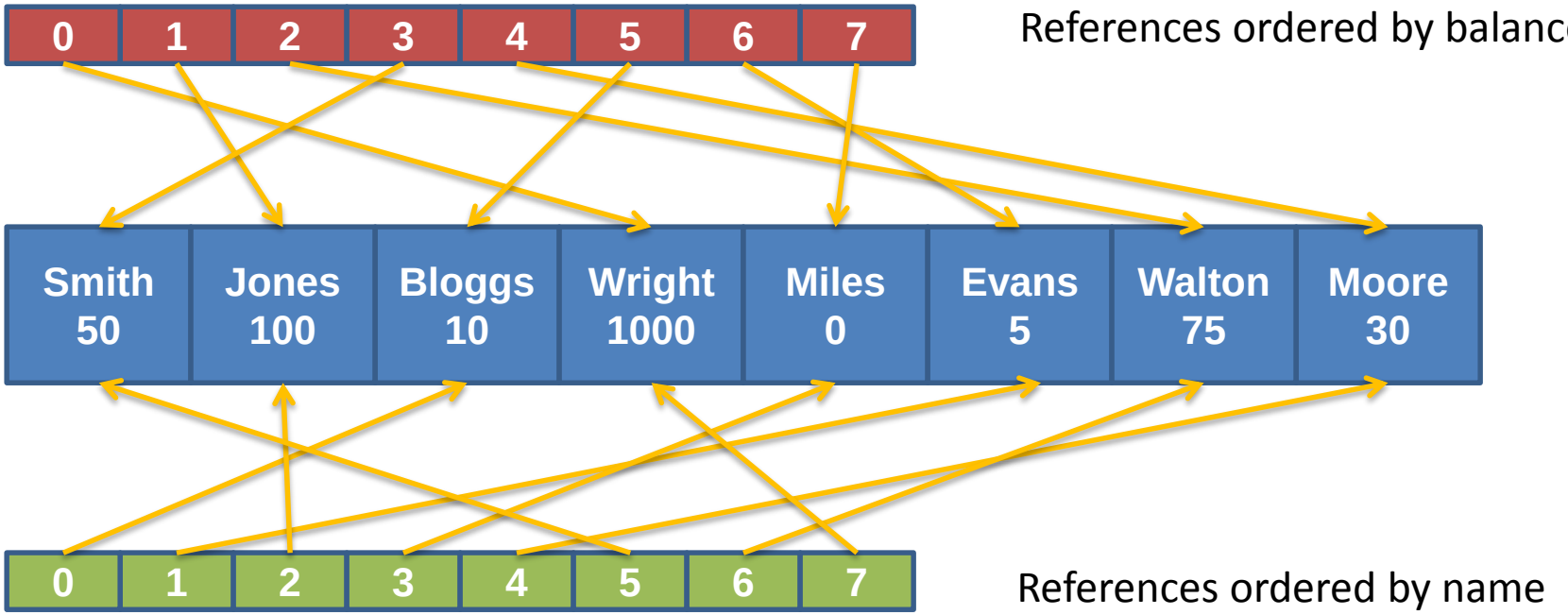
- When we store the data we could put the items in order
  - The data could be sorted in alphabetic order of account holder name
  - The data could be sorted in ascending order of bank balance
- However, it could not be sorted in two different orders at the same time
- To do that we would need to use two lists, which would be hard to keep up to date

Sorting using References



- Rather than sort the data itself, we can create a list of references which are sorted in a particular order
- The list of references above are sorted in order of bank balance

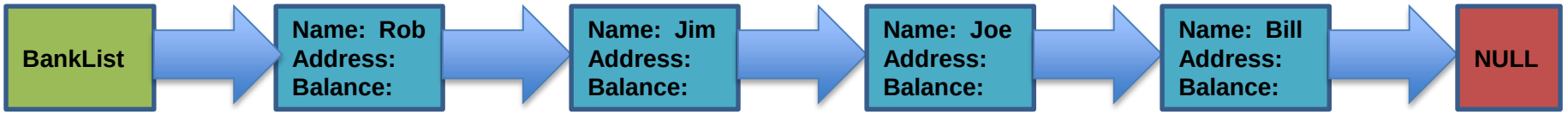
Sorting using References



- We can add another list of references to view the data sorted in a different order

The Linked Account Class

```
class Account
{
    public string Name;
    public int AccountNumber;
    public int Balance;
    public Account NextAccount;
}
Account BankList;
```



- If we put references inside our data values we can now create linked data structures
- This is useful because the storage can grow as required

Reference Power

- References are actually quite useful:
  - They allow us to manipulate large objects without moving them around in memory
  - They allow us to create multiple "views" of a set of data
  - They allow us to add extra objects up to the limits of the memory of the computer
  - They allow us to create data structures (lists, trees and meshes) in which data items are linked to others
- All in all, they are worth the effort