University of Hull

References and Arrays

Rob Miles

Department of Computer Science

29a 08120 Programming 2

Objects and References

- We now know that an object is a lump of data that sits in the memory of the computer
  - It lives at a particular location
- C# allows us to create references to objects in memory
- A reference is a tag which is tied to a particular object
  - A program can use the reference to find the object and do something with it
  - A single object can have multiple references referring to it
  - An object can have no references referring to it (which means that it may be garbage collected later)

An Array of Account References

```
class Account
{
    public string Name;
    public int AccountNumber;
    public int Balance;
}
Account [] BankAccounts = new Account [100] ;
```

- A program can create an array of references to objects
- Note that the above statements do **not** create any account storage
- Instead an array of references is created

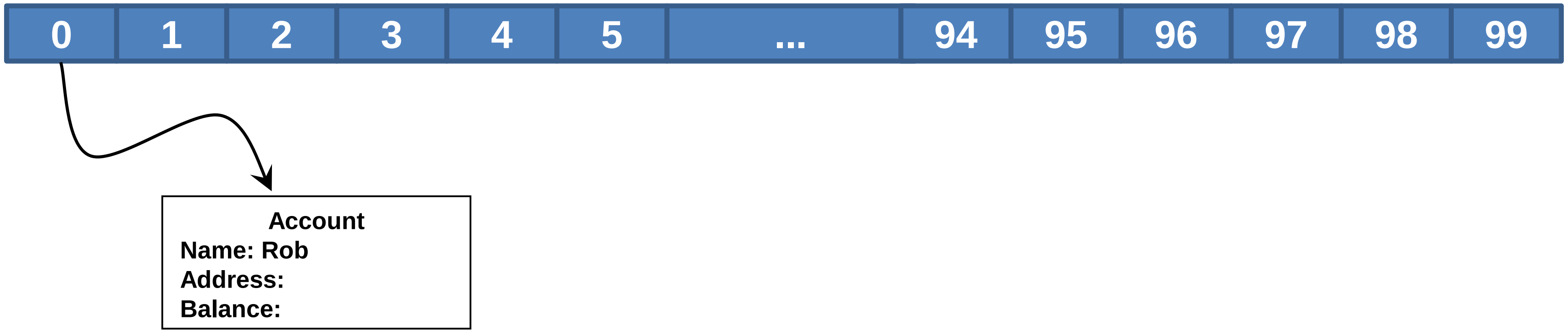
An Array of Account References

```
class Account
{
    public string Name;
    public int AccountNumber;
    public int Balance;
}
Account [] BankAccounts = new Account [100] ;
```

- It is very important to understand that the statements above have not created **any** `Account` storage
- Instead the statements have created 100 references that can refer to `Account` instances
- The program has to create the storage itself

Using a reference array

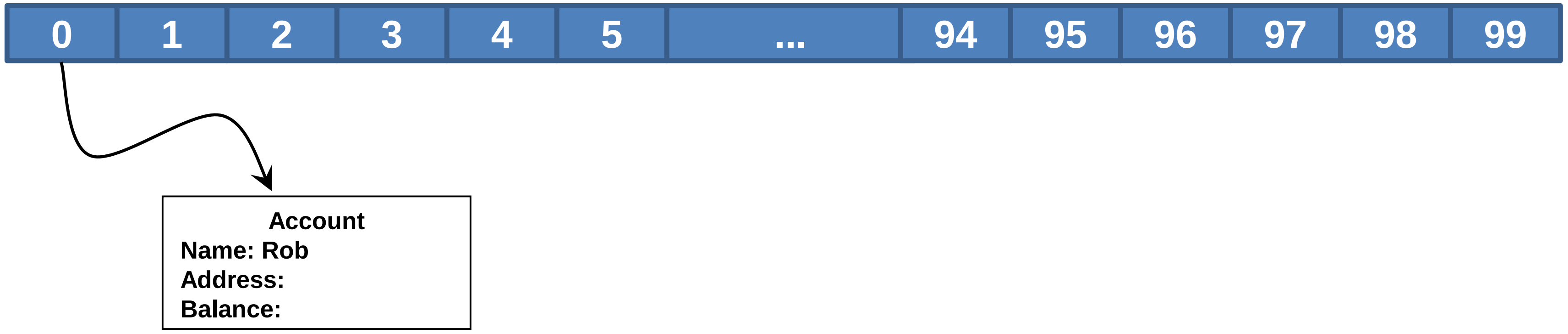
```
Account [] BankAccounts = new Account [100] ;
BankAccounts[0] = new Account();
BankAccounts[0].Name = "Rob";
```



- The above code sets up one element of the array
- The rest of the array elements all contain nothing

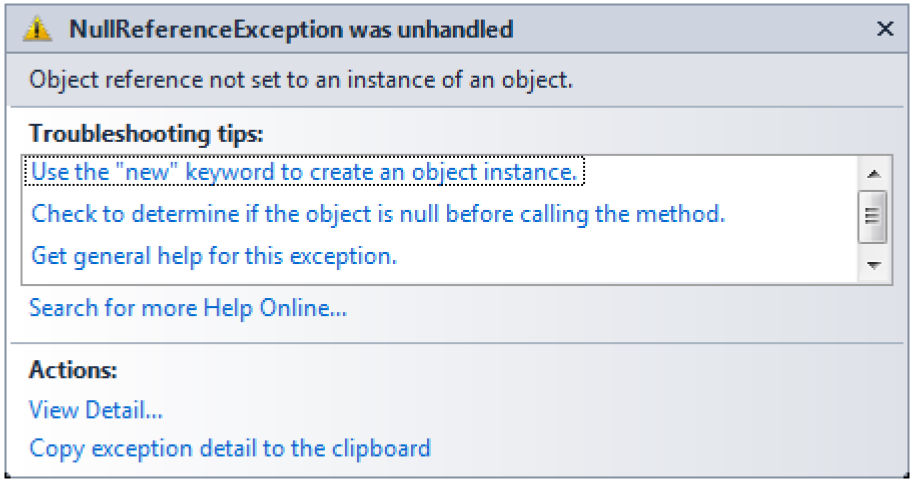
Reference array element problems

```
BankAccounts[1].Name = "Trevor";
```



- The above statement will fail
- Why?

Null references


NullReferenceException was unhandled

Object reference not set to an instance of an object.

Troubleshooting tips:

Use the "new" keyword to create an object instance.

Check to determine if the object is null before calling the method.

Get general help for this exception.

Search for more Help Online...

Actions:

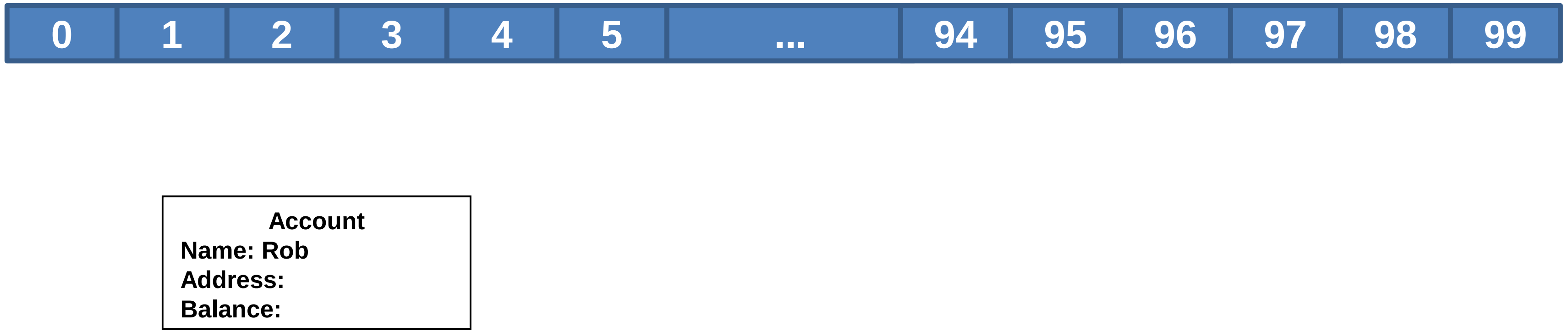
View Detail...

Copy exception detail to the clipboard

- When an array of references is created, each reference is set to the special value `null`
- If a program tries to follow a null reference the program will fail with a run time error

Setting a reference to null

```
BankAccounts[0] = null;
```



- The value `null` can be used in a program as a literal value
- If we set a reference to `null` it makes it point nowhere
- This might make work for the Garbage Collector

Comparing references with null

```
if (BankAccounts[0] == null)
    Console.WriteLine("Location is empty");
```

- A program can compare two references to determine if they refer to the same object
- If we compare a reference to `null` the comparison returns true if the reference is `null`
- A program could use this to find an empty location in an array of references

What does this code do?

```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

- This code is very useful/important
- But what does it do?

What does this code do?

```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

- This code will search through the accounts looking for one with the name "Rob“
- It is the code that runs inside your bank whenever you access your account

What does this code do?

```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

- Create a variable called `result`
- This is going to refer to the account that we find
- Initially it is set to `null`

What does this code do?

```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

- Set up a for loop to work through the elements in the `BankAccounts` array
- The `Length` property of the array tells the program how many elements it contains

What does this code do?

```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

- Test the name property to see if it matches the string `"Rob"`
- This is the account we are looking for

What does this code do?

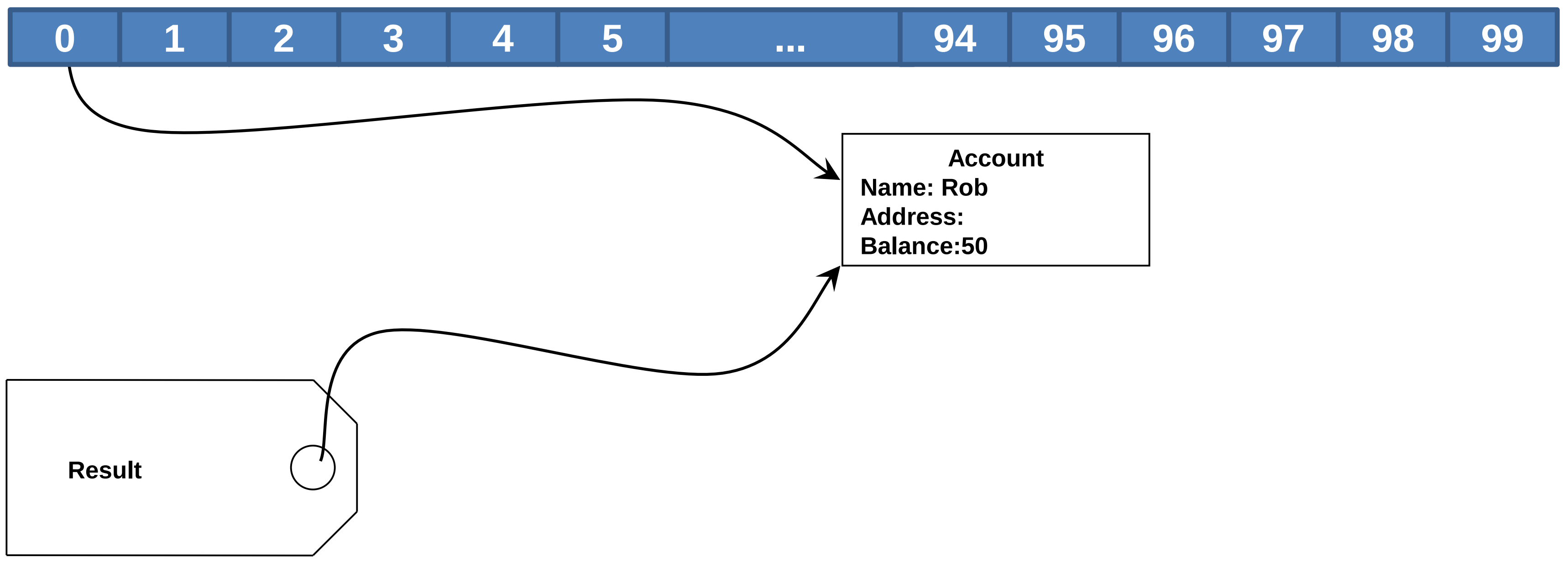
```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

- If the condition is true we make the result refer to the same object that the array element does

Setting the value of result

```
result = BankAccounts[i];
```



- If my account is at location 0 in the array we get the following arrangement

What does this code do?

```
Account result = null ;

for (int i = 0; i < BankAccounts.Length; i++)
{
    if (BankAccounts[i].Name == "Rob")
    {
        result = BankAccounts[i];
        break;
    }
}
```

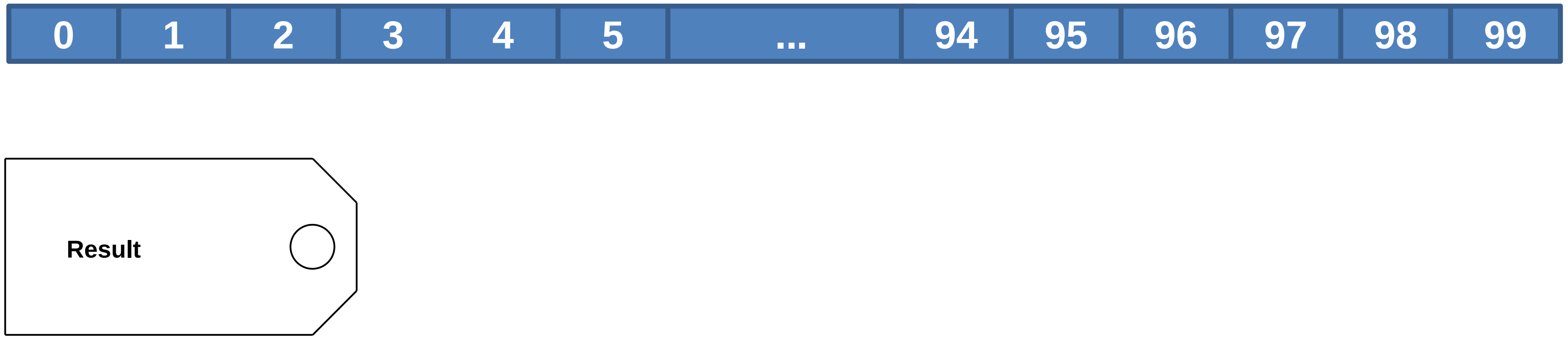
- This stops the loop looking any further if the account is found

Question

- What does it mean if we complete the loop and the variable result has the value `null` in it?

Question

- What does it mean if we complete the loop and the variable result has the value `null` in it?
- It means that there was no account with a matching name
- The result reference does not refer anywhere – it is `null`



Question

- What does it mean if we complete the loop and the variable result has the value `null` in it?
- It means that there was no account with a matching name
- We can test for this

```
if (result == null)
    Console.WriteLine("Account not found");
```

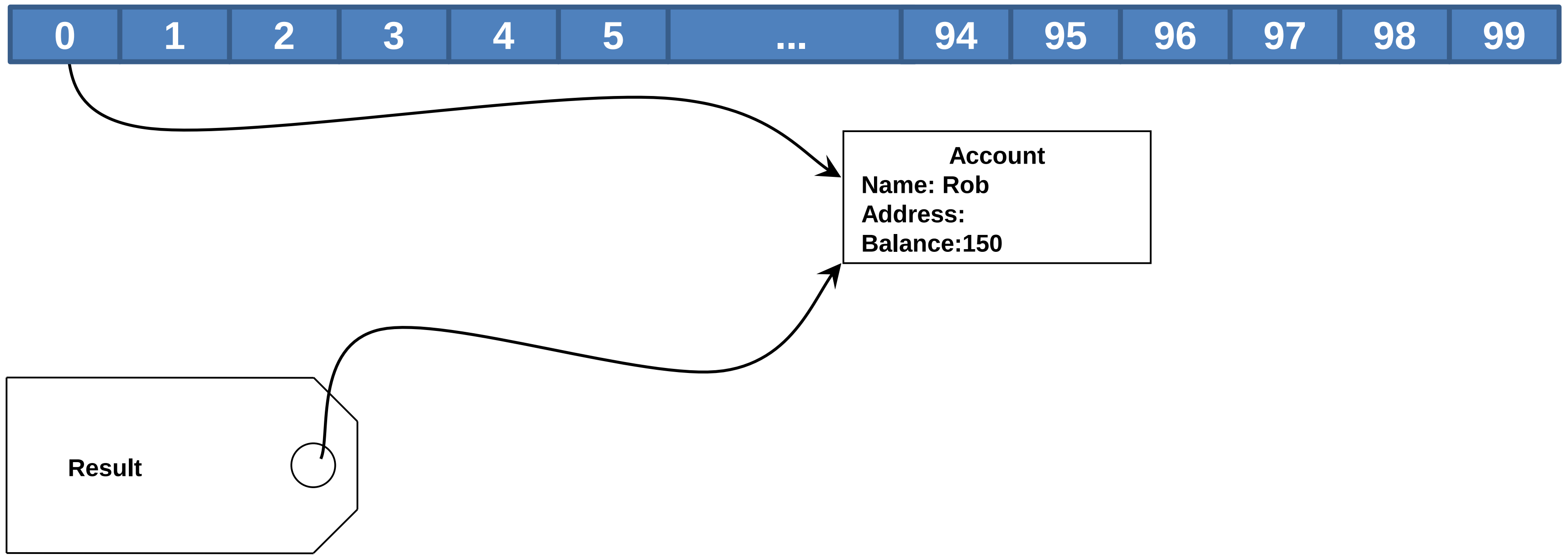
Using the result?

```
result.Balance = result.Balance + 100;
```

- A program can use the `result` reference to refer to the account that has been located
- The code above would give me 100 pounds
- There is no need to put the result "back" in the array as both the reference at `BankAccounts[0]` and the result reference both refer to the same object

Updating an object using a reference

```
result.Balance = result.Balance + 100;
```



- This is how the update will work

References and Arrays

- A program can create arrays that contain references to objects
- Initially all the elements in the array are set to null
  - This means that they do not refer to any object
- A program must create new instances of the objects and set the references in the array to refer to the instances
- Programs can search through arrays of references looking for items and returning a reference to the object that they find