Why objects and primitives different

Primitive variables are small and of known size

- They can be passed very quickly by variable
- Removed automatically with the frame
  Objects vary in size
- They are big and would be slow to pass around as parameters
- Their addresses can be passed quickly like primitive variables.
- Hang around in memory until garbage collection gets round to them

Primitives vs Objects

Stack

- Allocated as methods are called
- Deallocated when method exits
- Faster
- Limited in size
- Threadsafe – threads have their own stacks

Heap

- Allocated when objects are created
- Deallocated when garbage collector realises there is no link to object
- Slower
- Not limited in size
- Not threadsafe – the heap is shared

Instance vs Local Variables

```


public class simple
{
  private int num;

```

public bigger(int a) { int b = 3; num = a + b; } }

- **num** is an *instance variable*
  - It is created when the object is created
  - It is stored in the heap
- **b** is a *local variable*
  - It is created when the method is called
  - It is stored in the method frame
- **a** is also a *local variable*
- Remember the code itself is stored elsewhere in memory

```
</div>
```

Objects as "="

```


public main()
{
  simple s_1 = new simple(5);
  simple s_2 = new simple(7);
}
```

java heap example

```
</div>
```

Objects as "="

```


public main()
{
  simple s_1 = new simple(5);
  simple s_2 = new simple(7);
```

s_2 = s_1; }

- Assigning one object to another *copies the link* ***not*** *the object*

java heap example

```
</div>
```

Cloning an object

```


public main()
{
  simple s_1 = new simple(5);
```

simple s_2 = (simple) s_1.clone();

}

- Most Java classes come with a method called **clone()**
- This method returns a new object with the same values as the object is is called on
- You would need to create your own clone method for your own classes

java heap example

```
</div>
```

Objects - Testing Equality

```
public main()
{
  simple s_1 = new simple(5);
  simple s_2 = new simple(5);
```

if (s_1 == s_2) ... }

- This will fail, it will compare the two memory addresses and they will be different

java heap example

```
</div>
```

Objects - Testing Equality

```
public main()
{
  simple s_1 = new simple(5);
  simple s_2 = new simple(5);
```

if (s_1.equals(s_2)) ... }

- This will succeed
- Most Java classes implement equals()
- You will need to implement equals for your own classes
- Instructions on doing this

java heap example

```
</div>
```