

Inheritance - Classes & children; abstract

Avoid ambiguities - Java allows extending only 1 class

Interface: like an abstract class with no data, just specify some methods that are allowed

```
public interface MyInterface {
```

↑
name

// contract!

```
→ public void print() {
```

← input type, return type

// as many methods as wanted

```
}
```

```
public class MyClass extends MyParent implements MyInterface
```

if we want

↑

// Data

// Methods

```
→ public void print() {
```

// code appropriate to print this object

```
}
```

```
}
```

Example: Circular class

(L9-2)

Comparable interface $\rightarrow$ compareTo method

O_1 .compareTo(O_2) -1 if O_1 smaller
 0 if equal
 +1 if O_1 bigger

public abstract class Circular implements

Comparable<Circular>

// Circular objects among themselves

↑
what type of
object do you work on

// radius

// constant Pi

// constructor ...

public abstract double getArea();

public int compareTo(Circular c) {

if (this.getArea() < c.getArea()) return -1;

if (getArea() > c.getArea()) return +1;

return 0;

} // available in Circle, Sphere

// rewrite in Circle class if we want it different or
more specialized.

- - - Test Circular {
- - - main {

(L9-3)

```
Circle c = new Circle (1);  
Sphere s = new Sphere (2);  
System.out.println (c.compareTo(s));  
BankAccount b = new BankAccount (100);  
System.out.println (c.compareTo (b));  
                        ↑           ↑  
                  Circular   Circular
```

// Compilation error

// Test Circular file using packages

import CircularObjects;

↑
I want to use all classes defined in this package

import java.io; library (pre-defined classes & interfaces)

Read the java documentation!

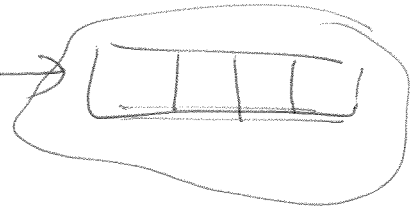
classes in a package have "privileged" access to each other public/private; no keyword → visible inside package not outside.

Comparable; Serializable

L9-4

double[] a;

a



content not address

Save in such a way that we can re-create object

public class MyClass implements Serializable {

look up in Java doc [↑] what methods
need to be specified

} // can implement other interfaces Comparable,
MyInterface...

Organizing code: packages

package CircularObjects;

public abstract class Circular {

} // Circular.java

package CircularObjects;

public class Circle extends Circular {

} // Circle.java

Sorting class :

Comparable <T> [] a;
 ↑
 generic Type

... a[i].compareTo(a[j]) ...

Java figures out what method to call here
 based on type of objects in a

Exceptions - errors

```
public void MyMethod() throws Exception {
}
```

Either pass it to caller or fix problem
 try/catch

```
try {
```

```
o.MyMethod();
```

```
} catch ( ) {
```

```
... code to deal with the exception }
```

Exceptions are classes like all other Java constructs
 public class MyException extends Exception {

customised error messages

}

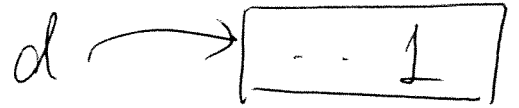
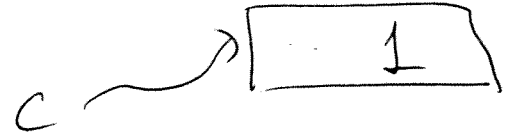
Comparing objects to determine if they are the same L9

Objects in java are references

Circle c, d;

c = new Circle(1);

d = new Circle(1);



if (c == d) {

↑ compares references, which are not same.

c = d;



Method equals

compare content, return true/false

if (c.equals(d))

}

// In Circle class

public boolean equals(Circle d) {

return radius == d.radius;

}