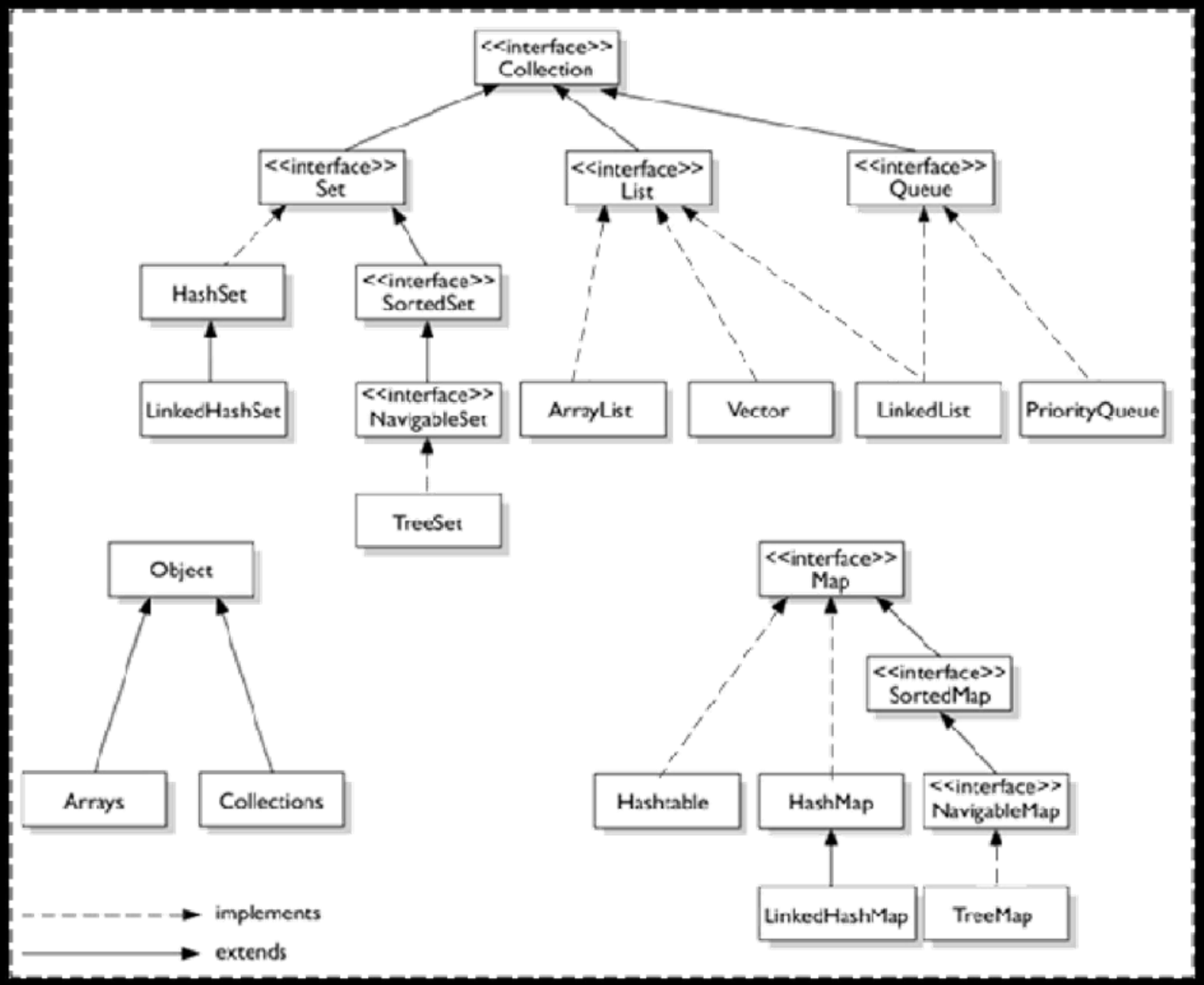
Java

Overview

- Object Oriented
  - Everything is object
  - All objects extend Object
- Platform Independent
- Static type
- Secure
  - There is no pointer
  - Garbage Collector
- Dynamic Class Loading
  - Used in web development

Java Collections



- all objects are passing by reference (primitive types by value)
- Array is fixed size like c++

  Definition: Datatype[ ] arrayName = new DataType[10]

  Access: arrayName[5]

- Array list is flexible size
  - Definition: arrayList<String> al = new ArrayList();
  - access: al.get(1)
- all primitive types have a class with capital initial
  - Int => Integer or float => Float
- For checking equality of objects should use .equal

```
String a = "DATA_STRUCTURE"
String b = "JAVA_CLASS"
if(a.equals(b)){.....}
```

- multiple arguments polygonForm(point… corners) { return corners[0]+corners[1]}

Package

- Packages are used in java in order to:
  - prevent naming conflict
  - control access
  - make searching and locating classes and interfaces easier
- if don't write any package, file will be in unnamed or default package and cannot be imported
  - just for small codes and testing
- packages in java is corresponds with file management
- It's reverse version of your company domain
  - package com.oracle.api
  - com/oracle/api
- Create file structures according to packages and using following command build .class files in separate folder
  - javac -sourcepath src -d bin src/ir/ds/javaproject/*.java

- Naming Convention

Class name
Interface name → Should start with uppercase letter

Method name
Variable name → Should start with lowercase letter

Package name → All letters lowercase

Constants name → All letters uppercase

- Access Levels

	class	Package	Subclass
Public	Y	Y	Y
Protected	Y	Y	Y
no modifier	Y	Y	N
Private	Y	N	N

Inheritance

- Doesn’t have multiple inheritance
  - `class A extends B {….}`
  - using super you have access to the parent class.
  - In the constructor of children should call super.
- Interface
  - `class A implements interface1, interface2 {….}`
  - All methods are virtual
  - All fields are final static implicitly
  - solves multiple inheritance problem

Hello world

HelloWorld.java:

Name of public class and file should be same

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

Command line

javac HelloWorld.java	Checking compile errors and generating bytecode
java HelloWorld	Executing ByteCode with JVM (java virtual machine)

```java
abstract class Human{
  protected String fname;
  public String surname;
  String city;

  public Human(String fname, String
    surname){ this.fname = fname;
    this.surname = surname;
  }

  public String getName(){
    return this.fname;
  }

  abstract double getSalary();

}
```

this is not a pointer

Needs implementation

```java
public class Student extends Human{

  private double GPA;


  public Student(String fname, String surname, double GPA){
    super(fname, surname);
    this.GPA = GPA;
  }

  public static void main(String[] args){
    Student s = new Student("fname","sname",17.01);
    System.out.println(s.getName());

  }

  @Override
  public String getName(){
    return GPA + " " + fname;
  }

  double getSalary(){
      return GPA*1.2;
  }

}
```

super is the parent

you can override all methods of parent using Override

```
interface haveProject  {

public String getProjectTitle();

public String getAdvisorName();

}
```

methods of all interfaces
don't have implementations

```
public class Student extends Human implements haveProject{

  @Override
  public String getProjectTitle(){
    return "COMPARING_DS_SHORTEST_PATH_ALGORITHM";
  }

  @Override
  public String getAdvisorName(){
    return "Advisor";
```

Implementation of interface