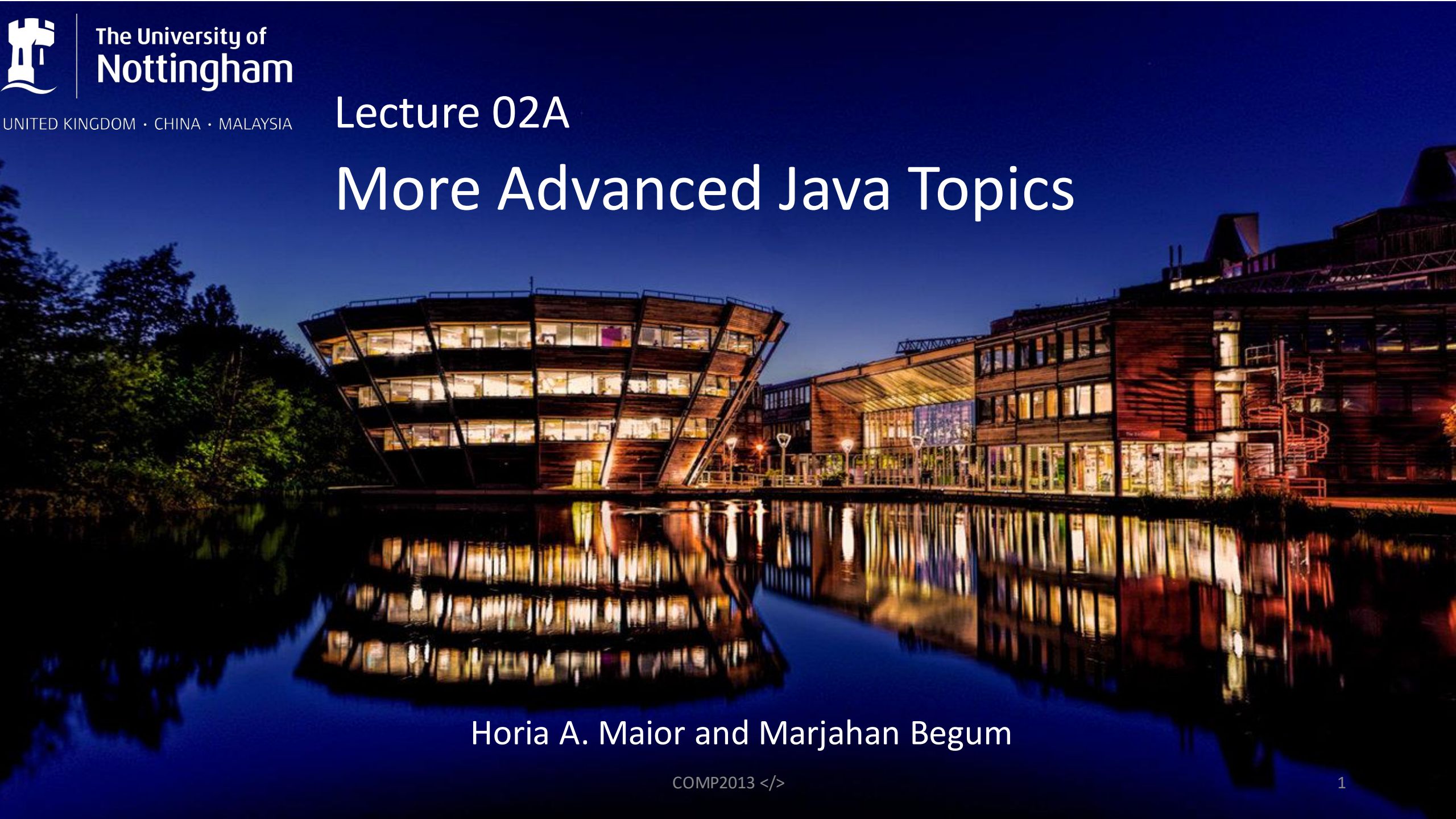
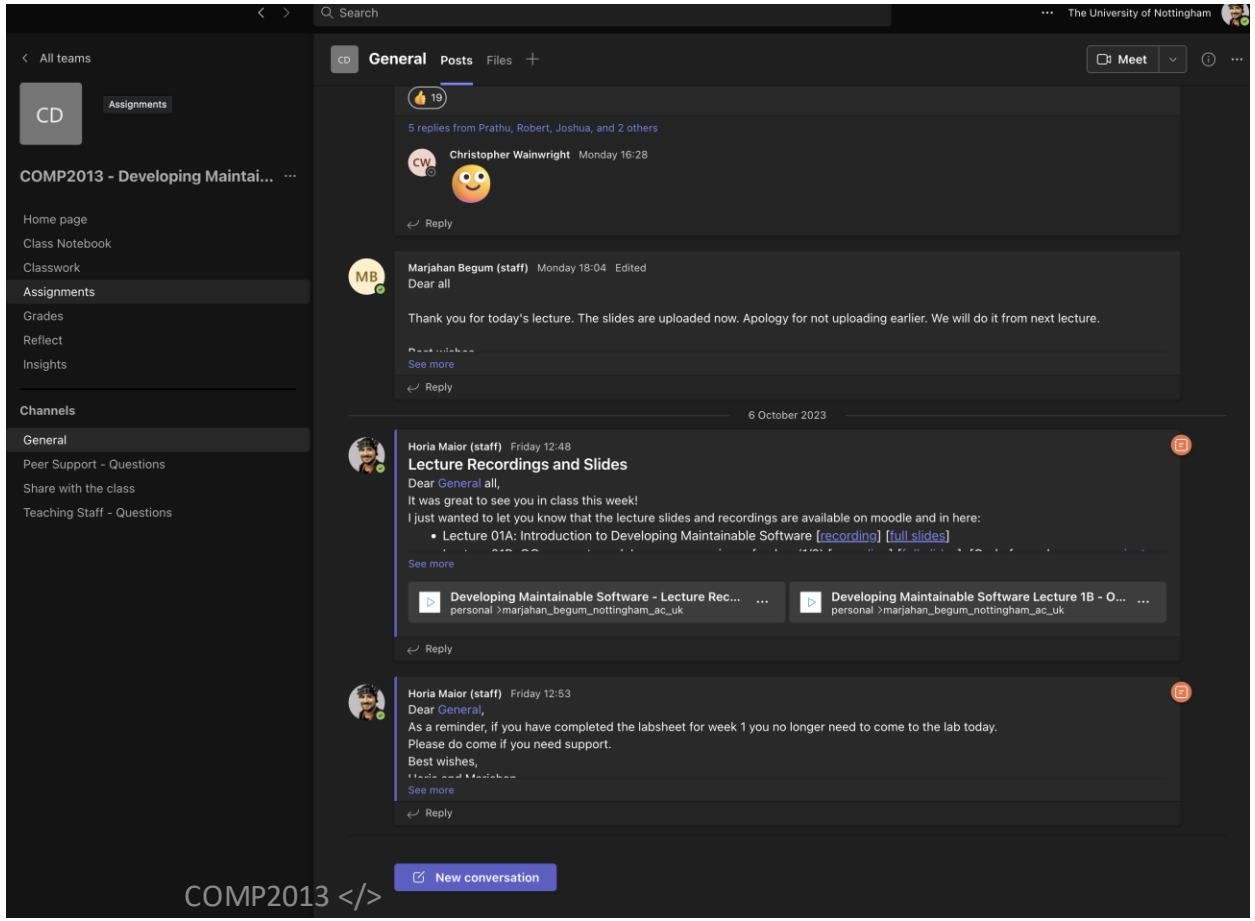


Lecture 02A

More Advanced Java Topics

Horia A. Maior and Marjahan Begum


Search
The University of Nottingham
All teams
CD
Assignments
COMP2013 - Developing Maintai...
Home page
Class Notebook
Classwork
Assignments
Grades
Reflect
Insights
Channels
General
Peer Support - Questions
Share with the class
Teaching Staff - Questions
General
Posts
Files
Meet
19
5 replies from Prathu, Robert, Joshua, and 2 others
Christopher Wainwright Monday 16:28
Reply
Marjahan Begum (staff) Monday 18:04 Edited
Dear all
Thank you for today's lecture. The slides are uploaded now. Apology for not uploading earlier. We will do it from next lecture.
See more
Reply
6 October 2023
Horia Maior (staff) Friday 12:48
Lecture Recordings and Slides
Dear General all,
It was great to see you in class this week!
I just wanted to let you know that the lecture slides and recordings are available on moodle and in here:
Lecture 01A: Introduction to Developing Maintainable Software [recording] [full slides]
See more
Developing Maintainable Software - Lecture Rec...
personal >marjahan_begum_nottingham_ac_uk
Developing Maintainable Software Lecture 1B - O...
personal >marjahan_begum_nottingham_ac_uk
Reply
Horia Maior (staff) Friday 12:53
Dear General,
As a reminder, if you have completed the labsheet for week 1 you no longer need to come to the lab today.
Please do come if you need support.
Best wishes,
See more
Reply
New conversation

Please note that the slides published AFTER the lectures are the official slides and are the ones that should be used for revision.

COMP2013

Developing Maintainable Software

Lecture 02A

More Advanced Java Topics

Horia A. Maior and Marjahan Begum

University of Nottingham
UK | CHINA | MALAYSIA

Topics for this Week

- Lecture 02A:
  - OO and Java Refresher (2/2)
    - The missing bits from last lecture
    - Java Collections framework
    - Implementation of object oriented concepts in Java
- Lab 02:
  - Implementing the ZooApp
- Lecture 02B:
  - IDEs + Java Releases
  - jShell
  - Maintaining the ZooApp (basic maintenance)

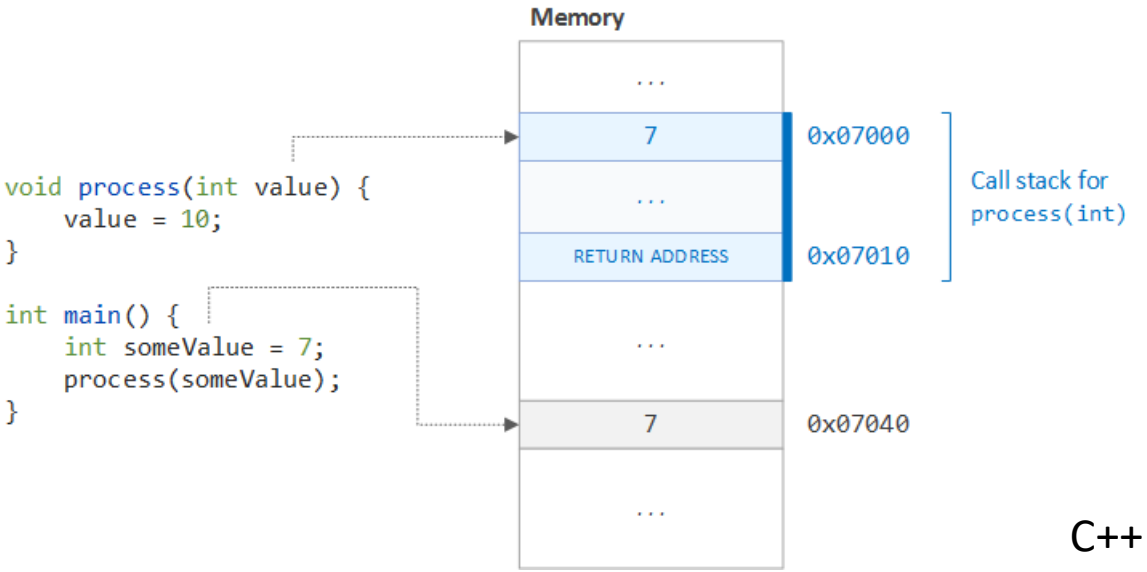
Learning Outcomes for this Week

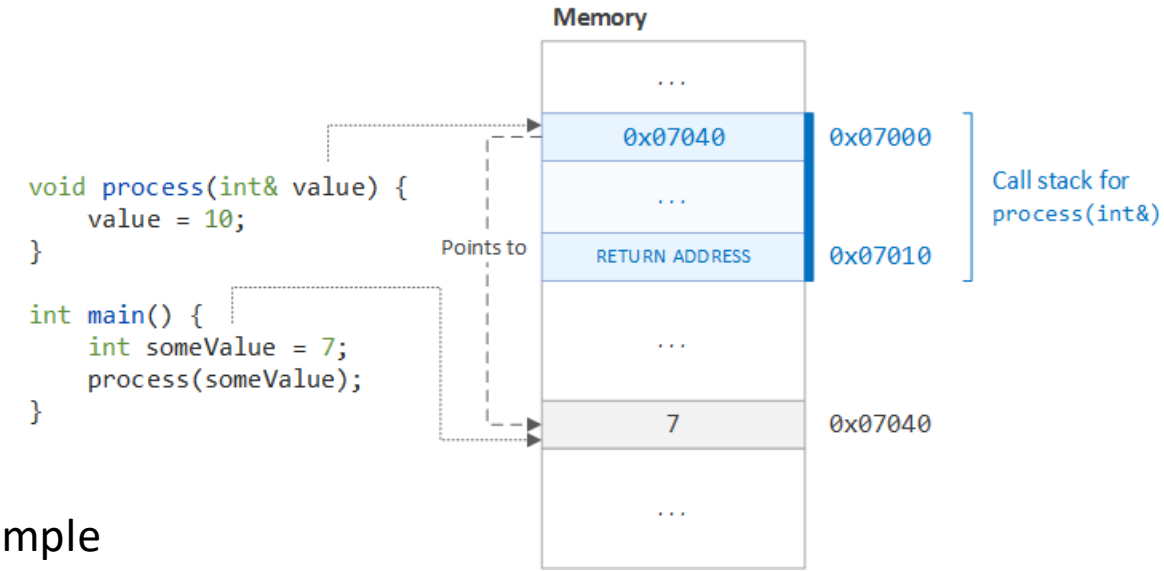
- At the end of this week …
  - You should have a foundational knowledge of Java
  - You should be able to implement all types of relationships and concepts
  - You should have a first idea about approaching basic software maintenance

the missing bits from last lecture

The Principle: Passing by Value vs Passing by Reference

- Passing by value constitutes copying of data, where changes to the copied value are not reflected in the original value
- Passing by reference constitutes the aliasing of data, where changes to the aliased value are reflected in the original value



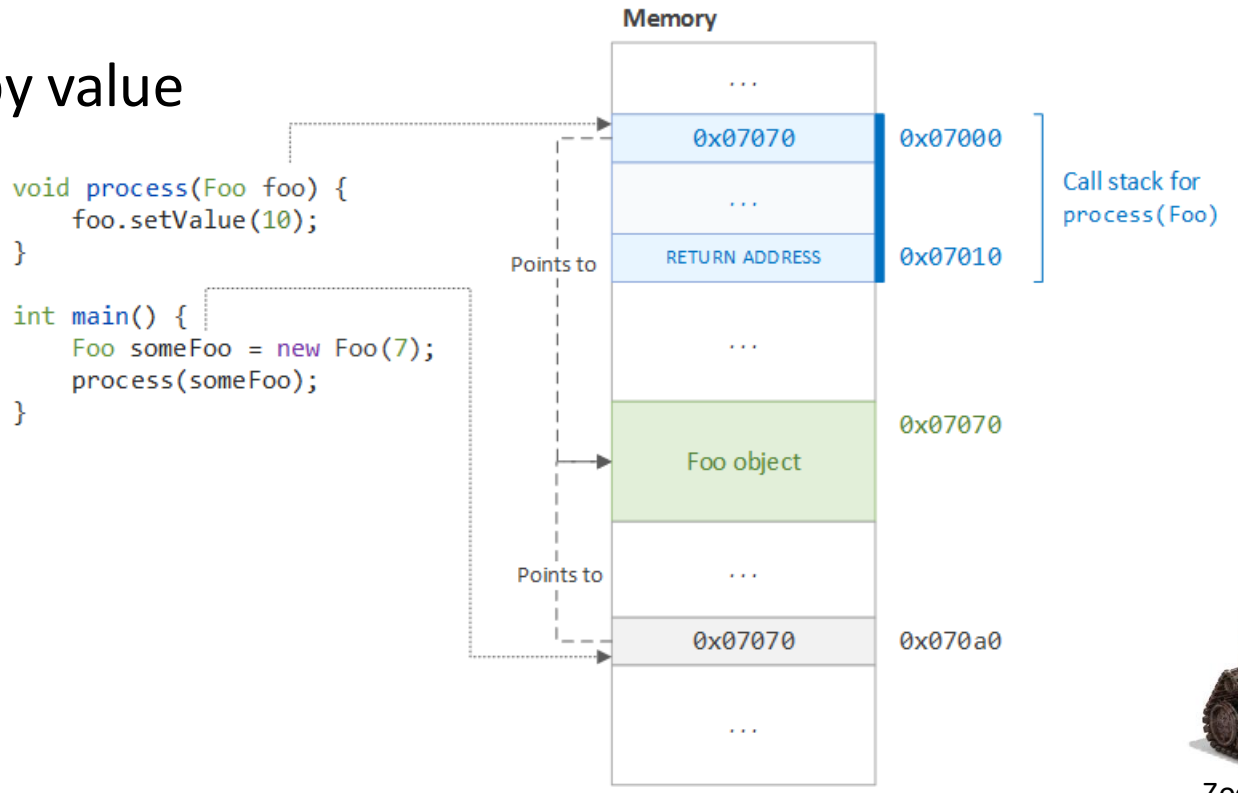


C++ Example

For more details see: https://dzone.com/articles/pass-by-value-vs-reference-in-java

Passing Data in Java: Always by Value!

- The value associated with an object is actually a pointer, called an "object reference" to the object in memory
- Object references are passed by value



ZooProjectStage5

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage5 - ZooApp.java

ZooProjectStage5 〉 src 〉 com 〉 Hagenbeck 〉 ZooApp

ZooApp

Project

ZooProjectStage5 E:\Teaching\COMP20
- .idea
- out
- src
  - com.Hagenbeck
    - Manager
    - Zoo
    - ZooApp
  - module-info.java
- ZooProjectStage5.iml
- External Libraries
- Scratches and Consoles

ZooApp.java | Zoo.java | Manager.java

```
1  package com.Hagenbeck;
2
3  public class ZooApp {
4
5      public static void main(String[] args) {
6          int avgVisitors=100;
7
8          Zoo zoo1; // creating reference
9          zoo1=new Zoo( location: "Hamburg");
10         zoo1=new Zoo( location: "Munic"); // new object takes over reference; old object not accessible any more
11         Zoo zoo2=zoo1; // references zoo1 and zoo2 point to same object
12         Zoo zoo3=new Zoo();
13         System.out.println("\nZoo1:"+zoo1);
14         System.out.println("Zoo2:"+zoo2);
15         System.out.println("Zoo3:"+zoo3);
16         zoo3.setLocation("Berlin");
17         zoo1.setLocation("Berlin"); // sets it for zoo1 and zoo2
18         System.out.println("\nZoo1:"+zoo1);
19         System.out.println("Zoo2:"+zoo2);
20         System.out.println("Zoo3:"+zoo3);
21         zoo1=new Zoo( location: "SanDiego"); // sets it for zoo1 only
22         System.out.println("\nZoo1:"+zoo1);
23         System.out.println("Zoo2:"+zoo2);
24         System.out.println("Zoo3:"+zoo3);
25
26         zoo1.changeZoo(zoo1, avgVisitors); // passing object reference and primitive
27         System.out.println("\n"+zoo1+":"+avgVisitors);
28     }
29 }
30
31
```

1 2

Structure Favorites

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure...

30:1 CRLF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage5 - Zoo.java

ZooProjectStage5 〉 src 〉 com 〉 Hagenbeck 〉 Zoo

ZooApp

Project

ZooApp.java | Zoo.java | Manager.java

```
ZooProjectStage5 E:\Teaching\COMP20
  .idea
  out
  src
    com.Hagenbeck
      Manager
      Zoo
      ZooApp
    module-info.java
  ZooProjectStage5.iml
External Libraries
Scratches and Consoles
```

```
1  package com.Hagenbeck;
2
3  public class Zoo {
4
5      private String location;
6
7      public Zoo() { this( location: "Unknown"); }
10
11     public Zoo(String location) {
12         //this.location=location;
13         this.setLocation(location);
14         System.out.println("\nConstructing "+this);
15     }
16
17     public void doSomething() { Manager manager=new Manager(); }
20
21     public String getLocation() { return location; }
24
25     public void setLocation(String location) { this.location = location; }
28     public void changeZoo(Zoo zoo, int avgVisitors) {
29         avgVisitors=200;
30         System.out.println("\n@Local Start:"+zoo);
31         zoo=new Zoo( location: "London");
32         //zoo.setLocation("Munic");
33         //this.setLocation("Amsterdam");
34         System.out.println("\n@Local End:"+zoo);
35     }
36     @Override
37     public String toString() { return getClass().getSimpleName()+"("+this.getLocation()+")"; }
40 }
41
```

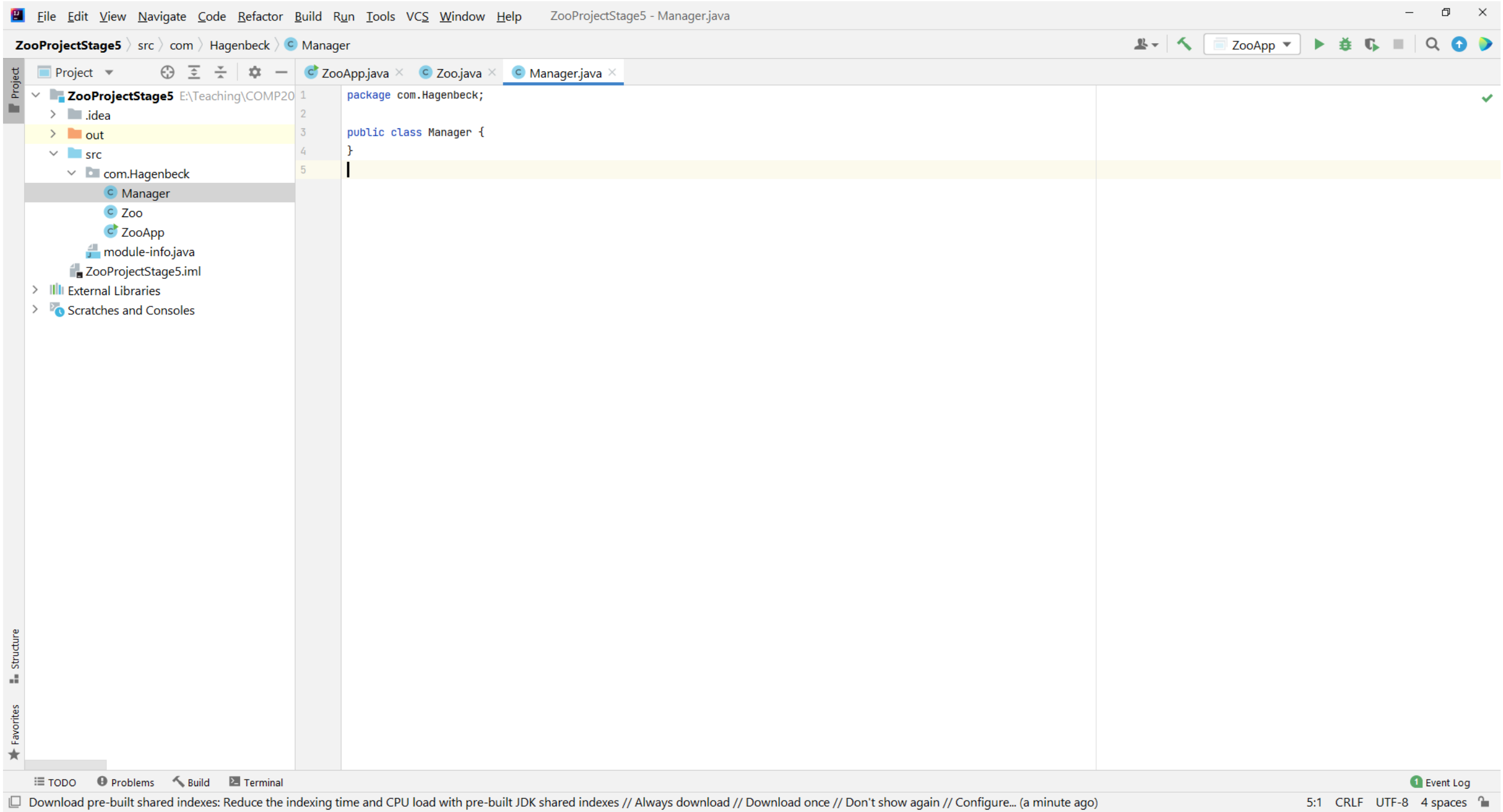
4 1 1

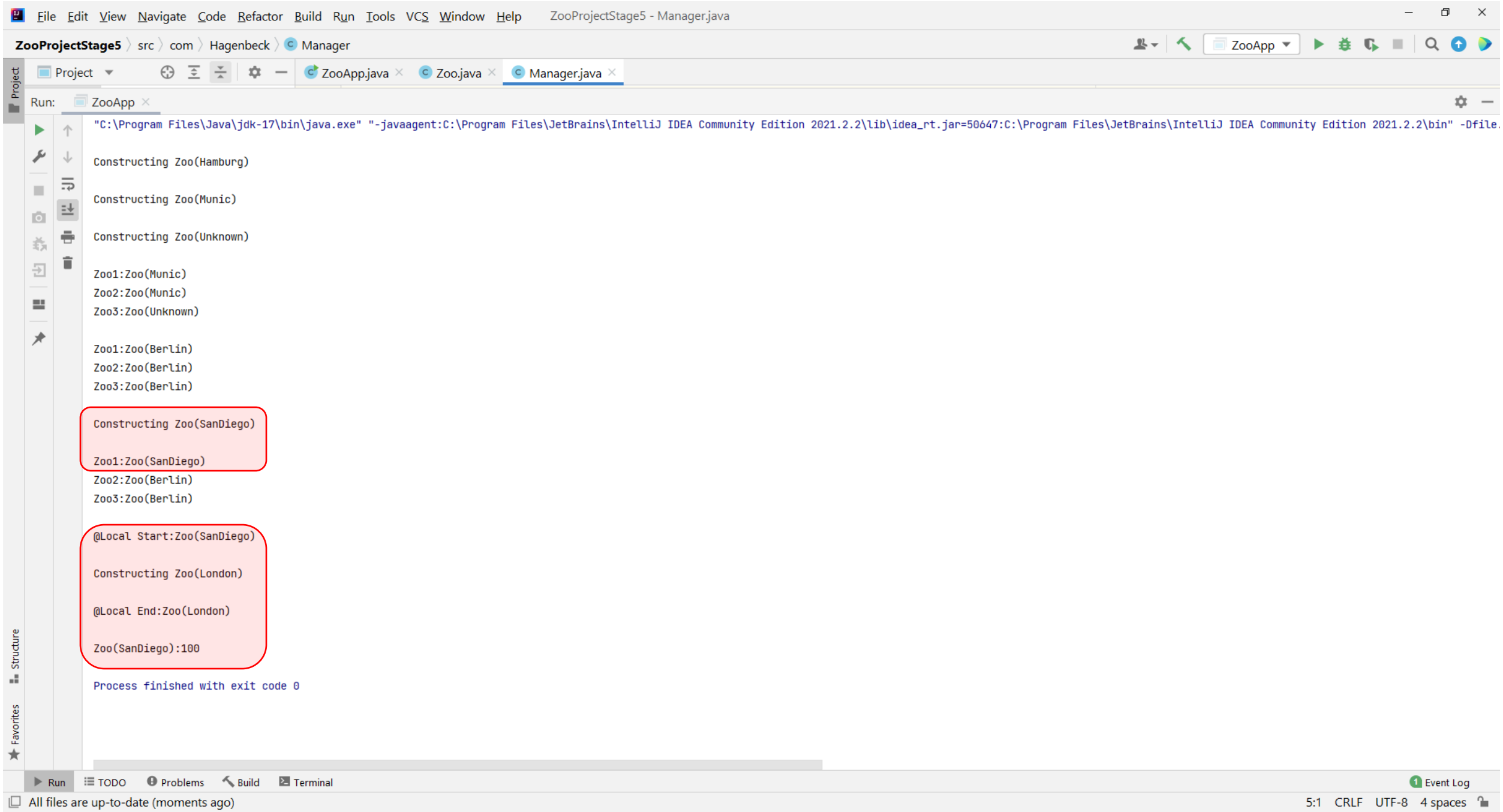
Structure Favorites

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure...

41:1 CRLF UTF-8 Tab*


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ZooProjectStage5 - Manager.java
ZooProjectStage5 〉 src 〉 com 〉 Hagenbeck 〉 Manager
ZooApp
Project
ZooProjectStage5 E:\Teaching\COMP20
.idea
out
src
com.Hagenbeck
Manager
Zoo
ZooApp
module-info.java
ZooProjectStage5.iml
External Libraries
Scratches and Consoles
ZooApp.java
Zoo.java
Manager.java
1 package com.Hagenbeck;
2
3 public class Manager {
4 }
5
Structure
Favorites
TODO
Problems
Build
Terminal
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago)
5:1 CRLF UTF-8 4 spaces


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ZooProjectStage5 - Manager.java
ZooProjectStage5 src com Hagenbeck Manager
ZooApp
Project
ZooApp.java
Zoo.java
Manager.java
Run: ZooApp
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\lib\idea_rt.jar=50647:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\bin" -Dfile.
Constructing Zoo(Hamburg)
Constructing Zoo(Munic)
Constructing Zoo(Unknown)
Zoo1:Zoo(Munic)
Zoo2:Zoo(Munic)
Zoo3:Zoo(Unknown)
Zoo1:Zoo(Berlin)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)
Constructing Zoo(SanDiego)
Zoo1:Zoo(SanDiego)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)
@Local Start:Zoo(SanDiego)
Constructing Zoo(London)
@Local End:Zoo(London)
Zoo(SanDiego):100
Process finished with exit code 0
Run TODO Problems Build Terminal
Event Log
All files are up-to-date (moments ago)
5:1 CRLF UTF-8 4 spaces

More Hacks

- Using "this" inside methods
  - Inside a method, "this" refers to the object for which the method was called. "this" can be passed to other constructors and methods as a parameter.

- Return statements in void methods
  - A void method can use a return statement to quit the method early
  - There is no need for a return at the end

ZooProjectStage6

Stage 6

- Lets Create and add:
- A "Manager" object (and passing "this" as parameter) in "doSomething" method in "Zoo" class and
- adding "sayHello" method call;
- "doSomething" method call to "Zoo" class constructor;
- constructor to "Manager" class;
- add "sayHello" method with "return" statement (to stop method execution) to "Manager" class

ZooProjectStage6

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage6 - ZooApp.java

ZooProjectStage6 〉 src 〉 com 〉 Hagenbeck 〉 ZooApp

ZooApp

Project

ZooProjectStage6 E:\Teaching\COMP2013-G52DMS\G52D
- .idea
- out
- src
  - com.Hagenbeck
    - Manager
    - Zoo
    - ZooApp
  - module-info.java
- ZooProjectStage6.iml
- External Libraries
- Scratches and Consoles

ZooApp.java | Zoo.java | Manager.java

```
1  package com.Hagenbeck;
2
3  public class ZooApp {
4
5      public static void main(String[] args) {
6          int avgVisitors=100;
7
8          Zoo zoo1; // creating reference
9          zoo1=new Zoo( location: "Hamburg");
10         zoo1=new Zoo( location: "Munic"); // new object takes over reference; old object not accessible any more
11         Zoo zoo2=zoo1; // references zoo1 and zoo2 point to same object
12         Zoo zoo3=new Zoo();
13         System.out.println("\nZoo1:"+zoo1);
14         System.out.println("Zoo2:"+zoo2);
15         System.out.println("Zoo3:"+zoo3);
16         zoo3.setLocation("Berlin");
17         zoo1.setLocation("Berlin"); // sets it for zoo1 and zoo2
18         System.out.println("\nZoo1:"+zoo1);
19         System.out.println("Zoo2:"+zoo2);
20         System.out.println("Zoo3:"+zoo3);
21         zoo1=new Zoo( location: "SanDiego"); // sets it for zoo1 only
22         System.out.println("\nZoo1:"+zoo1);
23         System.out.println("Zoo2:"+zoo2);
24         System.out.println("Zoo3:"+zoo3);
25
26         zoo1.changeZoo(zoo1, avgVisitors); // passing object reference and primitive
27         System.out.println("\n"+zoo1+":"+avgVisitors);
28     }
29 }
30
31
```

Structure Favorites

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (moments ago) 3:14 CRLF UTF-8 Tab*

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage6 - Zoo.java

ZooProjectStage6 〉 src 〉 com 〉 Hagenbeck 〉 Zoo

ZooApp

Project

ZooProjectStage6 E:\Teaching\COMP2013-G52DMS\G52D
- .idea
- out
- src
  - com.Hagenbeck
    - Manager
    - Zoo
    - ZooApp
  - module-info.java
- ZooProjectStage6.iml
- External Libraries
- Scratches and Consoles

ZooApp.java | Zoo.java | Manager.java

```java
package com.Hagenbeck;

public class Zoo {

    private String location;

    public Zoo() { this( location: "Unknown"); }

    public Zoo(String location) {
        //this.location=location;
        this.setLocation(location);
        System.out.println("\nConstructing "+this);
        doSomething();
    }

    public void doSomething() {
        Manager manager=new Manager( zoo: this);
        manager.sayHello();
    }

    public String getLocation() { return location; }

    public void setLocation(String location) { this.location = location; }
    public void changeZoo(Zoo zoo, int avgVisitors) {
        avgVisitors=200;
        System.out.println("\n@Local Start:"+zoo);
        zoo=new Zoo( location: "London");
        //zoo.setLocation("Munic");
        //this.setLocation("Amsterdam");
        System.out.println("\n@Local End:"+zoo);
    }
    @Override
    public String toString() { return getClass().getSimpleName()+"("+this.getLocation()+")"; }
}

```

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago) 43:1 CRLF UTF-8 Tab*

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage6 - Manager.java

ZooProjectStage6 〉 src 〉 com 〉 Hagenbeck 〉 Manager

ZooApp

Project

ZooProjectStage6 E:\Teaching\COMP2013-G52DMS\G52D
- .idea
- out
- src
  - com.Hagenbeck
    - Manager
    - Zoo
    - ZooApp
  - module-info.java
- ZooProjectStage6.iml
- External Libraries
- Scratches and Consoles

ZooApp.java | Zoo.java | Manager.java

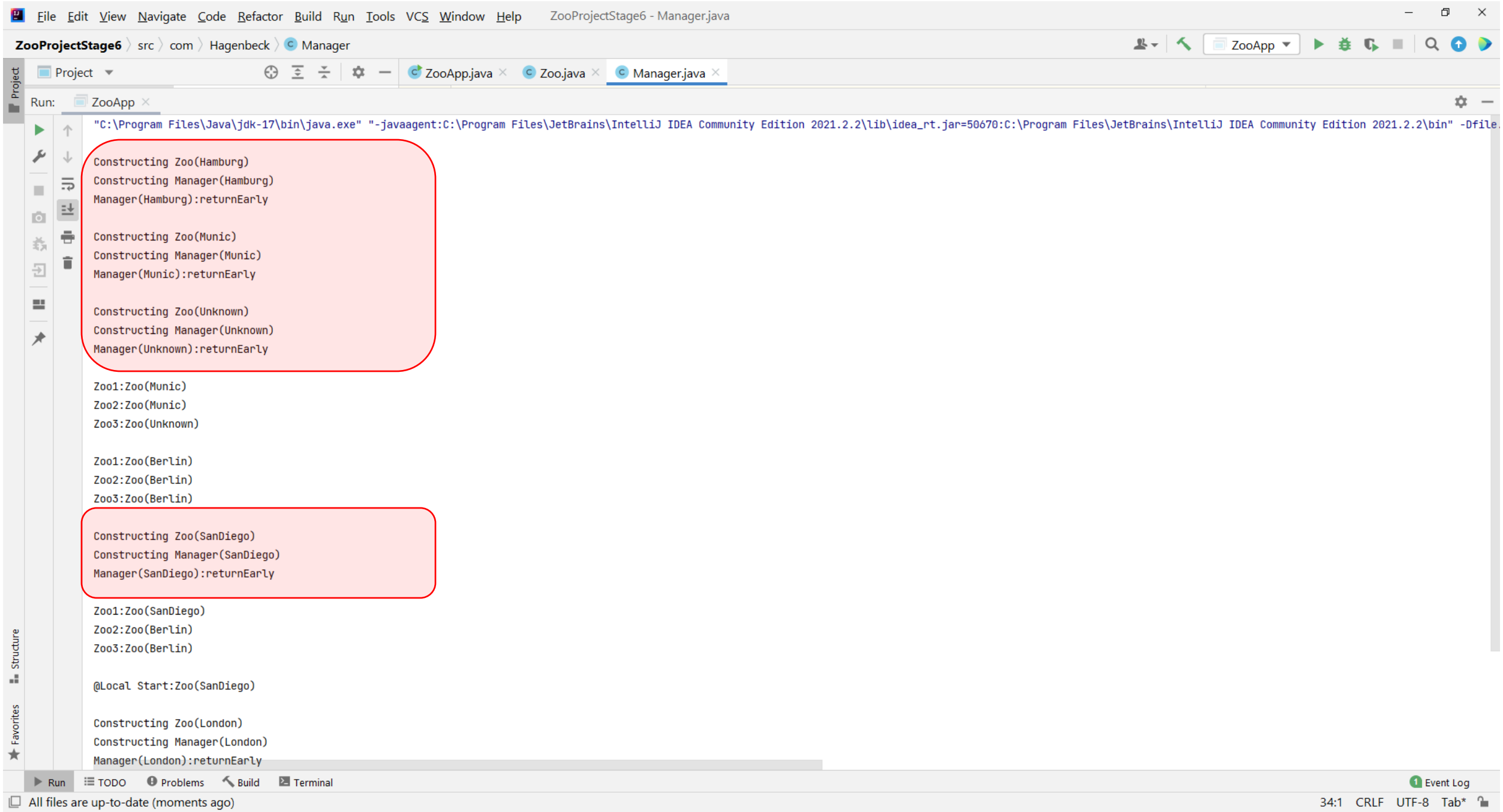
```
1  package com.Hagenbeck;
2
3  public class Manager {
4
5      private Zoo zoo;
6
7      public Manager(Zoo zoo) {
8          this.setZoo(zoo);
9          System.out.println("Constructing "+this);
10     }
11
12     public void sayHello() {
13         //boolean returnEarly=false;
14         boolean returnEarly=true;
15         if(returnEarly) {
16             System.out.println(this+":returnEarly");
17             return;
18         }
19         System.out.println(this+":returnLate");
20     }
21
22     public Zoo getZoo() { return zoo; }
25
26     public void setZoo(Zoo zoo) { this.zoo = zoo; }
29
30     public String toString() { return getClass().getSimpleName()+"("+zoo.getLocation()+")"; }
33 }
34
```

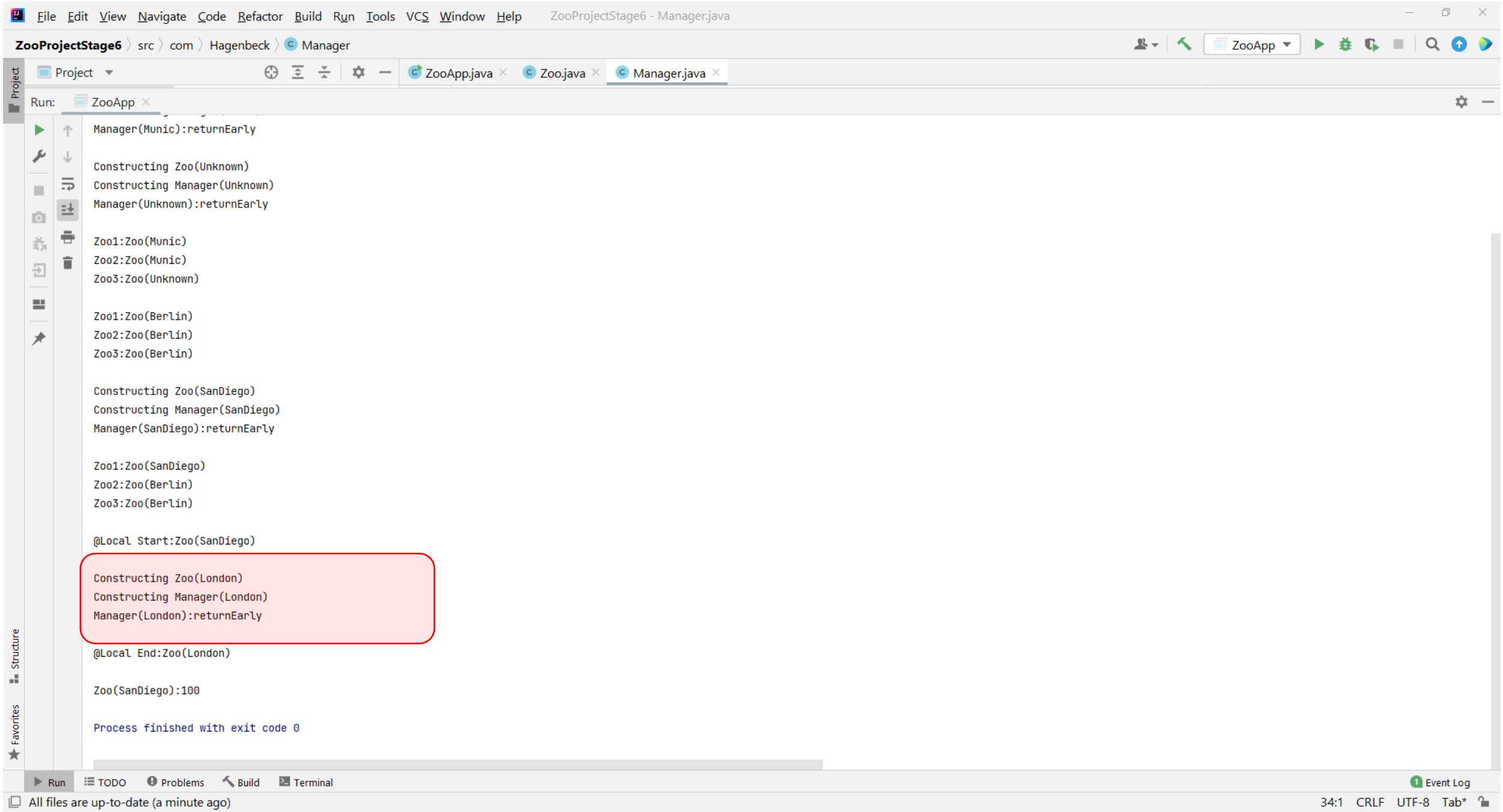
2

Structure Favorites

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago) 34:1 CRLF UTF-8 Tab*


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ZooProjectStage6 - Manager.java
ZooProjectStage6 〉 src 〉 com 〉 Hagenbeck 〉 Manager
ZooApp
Project
ZooApp.java
Zoo.java
Manager.java
Run: ZooApp
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\lib\idea_rt.jar=50670:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\bin" -Dfile
Constructing Zoo(Hamburg)
Constructing Manager(Hamburg)
Manager(Hamburg):returnEarly
Constructing Zoo(Munic)
Constructing Manager(Munic)
Manager(Munic):returnEarly
Constructing Zoo(Unknown)
Constructing Manager(Unknown)
Manager(Unknown):returnEarly
Zoo1:Zoo(Munic)
Zoo2:Zoo(Munic)
Zoo3:Zoo(Unknown)
Zoo1:Zoo(Berlin)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)
Constructing Zoo(SanDiego)
Constructing Manager(SanDiego)
Manager(SanDiego):returnEarly
Zoo1:Zoo(SanDiego)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)
@Local Start:Zoo(SanDiego)
Constructing Zoo(London)
Constructing Manager(London)
Manager(London):returnEarly
Run TODO Problems Build Terminal
Event Log
All files are up-to-date (moments ago)
34:1 CRLF UTF-8 Tab*


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ZooProjectStage6 - Manager.java
ZooProjectStage6 src com Hagenbeck Manager
ZooApp
Project
ZooApp.java
Zoo.java
Manager.java
Run: ZooApp
Manager(Munic):returnEarly
Constructing Zoo(Unknown)
Constructing Manager(Unknown)
Manager(Unknown):returnEarly
Zoo1:Zoo(Munic)
Zoo2:Zoo(Munic)
Zoo3:Zoo(Unknown)
Zoo1:Zoo(Berlin)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)
Constructing Zoo(SanDiego)
Constructing Manager(SanDiego)
Manager(SanDiego):returnEarly
Zoo1:Zoo(SanDiego)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)
@Local Start:Zoo(SanDiego)
Constructing Zoo(London)
Constructing Manager(London)
Manager(London):returnEarly
@Local End:Zoo(London)
Zoo(SanDiego):100
Process finished with exit code 0
Run TODO Problems Build Terminal
Event Log
All files are up-to-date (a minute ago)
34:1 CRLF UTF-8 Tab*

More Hacks

- Overloaded Methods
  - Methods of the **same class** that have the **same name** but **different numbers** or **types of parameters**
  - The compiler treats overloaded methods as completely different methods
  - Return Type Doesn't Matter
  - Compile-Time Resolution. This is known as compile-time polymorphism or static polymorphism.
  - The compiler knows which one to call based on the number and the types of the parameters passed to the method
  - The return type alone is not sufficient for making a distinction between overloaded methods

Static Fields

- A static field (class field or class variable) is shared by all objects of the class
- A non-static field (instance field or instance variable) belongs to an individual object
- Static fields are stored with the class code, separately from instance variables that describe an individual object
- **Modifying a Static Field Affects All Instances**

Static Fields

- Public static fields, usually global constants, are referred to in other classes using dot notation
  - ClassName.constName

- Usually static fields are NOT initialised in constructors; they are initialised either in declarations or in public static methods or just use their default value

- If a class has only static fields, there is no point in creating objects of that class, as all of them would be identical
  - Math and System are examples of the above (they have no public constructors and cannot be instantiated)

Static Methods

- Static methods can access and manipulate class's static fields. They belong to the class - not an instance of it.
- Static methods are called using dot notation
  - ClassName.statMethod(...)

- Note that static methods cannot access instance fields or call instance methods of the class while instance methods can access all fields and call all methods of their class - both static and non-static

- Static methods will usually take input from the parameters, perform actions on it, then return some result.

Static Methods

- When to use static methods (Stack Overflow)
  - https://stackoverflow.com/questions/2671496/when-to-use-static-methods
- Stage 7 (more coding):
  - Adding static field "numZoosCreated" to "Zoo" class;
  - increasing "numZoosCreated" in Zoo class constructor
  - adding "numZoosCreated" getter to Zoo class;
  - Check "numZoosCreated" in "ZooApp" class

ZooProjectStage7

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage7 - ZooApp.java

ZooProjectStage7 〉 src 〉 com 〉 Hagenbeck 〉 ZooApp

ZooApp

Project

ZooProjectStage7 E:\Teaching\COMP2013-G52DMS\G52D
- .idea
- out
- src
  - com.Hagenbeck
    - Manager
    - Zoo
    - ZooApp
  - module-info.java
- ZooProjectStage7.iml
- External Libraries
- Scratches and Consoles

ZooApp.java | Manager.java | Zoo.java

```
1  package com.Hagenbeck;
2
3  public class ZooApp {
4
5      public static void main(String[] args) {
6          int avgVisitors=100;
7
8          Zoo zoo1; // creating reference
9          zoo1=new Zoo( location: "Hamburg");
10         zoo1=new Zoo( location: "Munic"); // new object takes over reference; old object not accessible any more
11         Zoo zoo2=zoo1; // references zoo1 and zoo2 point to same object
12         Zoo zoo3=new Zoo();
13         System.out.println("\nZoo1:"+zoo1);
14         System.out.println("Zoo2:"+zoo2);
15         System.out.println("Zoo3:"+zoo3);
16         zoo3.setLocation("Berlin");
17         zoo1.setLocation("Berlin"); // sets it for zoo1 and zoo2
18         System.out.println("\nZoo1:"+zoo1);
19         System.out.println("Zoo2:"+zoo2);
20         System.out.println("Zoo3:"+zoo3);
21         zoo1=new Zoo( location: "SanDiego"); // sets it for zoo1 only
22         System.out.println("\nZoo1:"+zoo1);
23         System.out.println("Zoo2:"+zoo2);
24         System.out.println("Zoo3:"+zoo3);
25
26         zoo1.changeZoo(zoo1, avgVisitors); // passing object reference and primitive
27         System.out.println("\n"+zoo1+":"+avgVisitors);
28         System.out.println("\ngetNumZoosCreated(): "+Zoo.getNumZoosCreated());
29     }
30 }
31
32
```

1 2

Structure Favorites

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't sh... (moments ago) Checking for JDK updates 31:1 CRLF UTF-8 Tab*

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage7 - Zoo.java

ZooProjectStage7 〉 src 〉 com 〉 Hagenbeck 〉 Zoo

ZooApp

Project

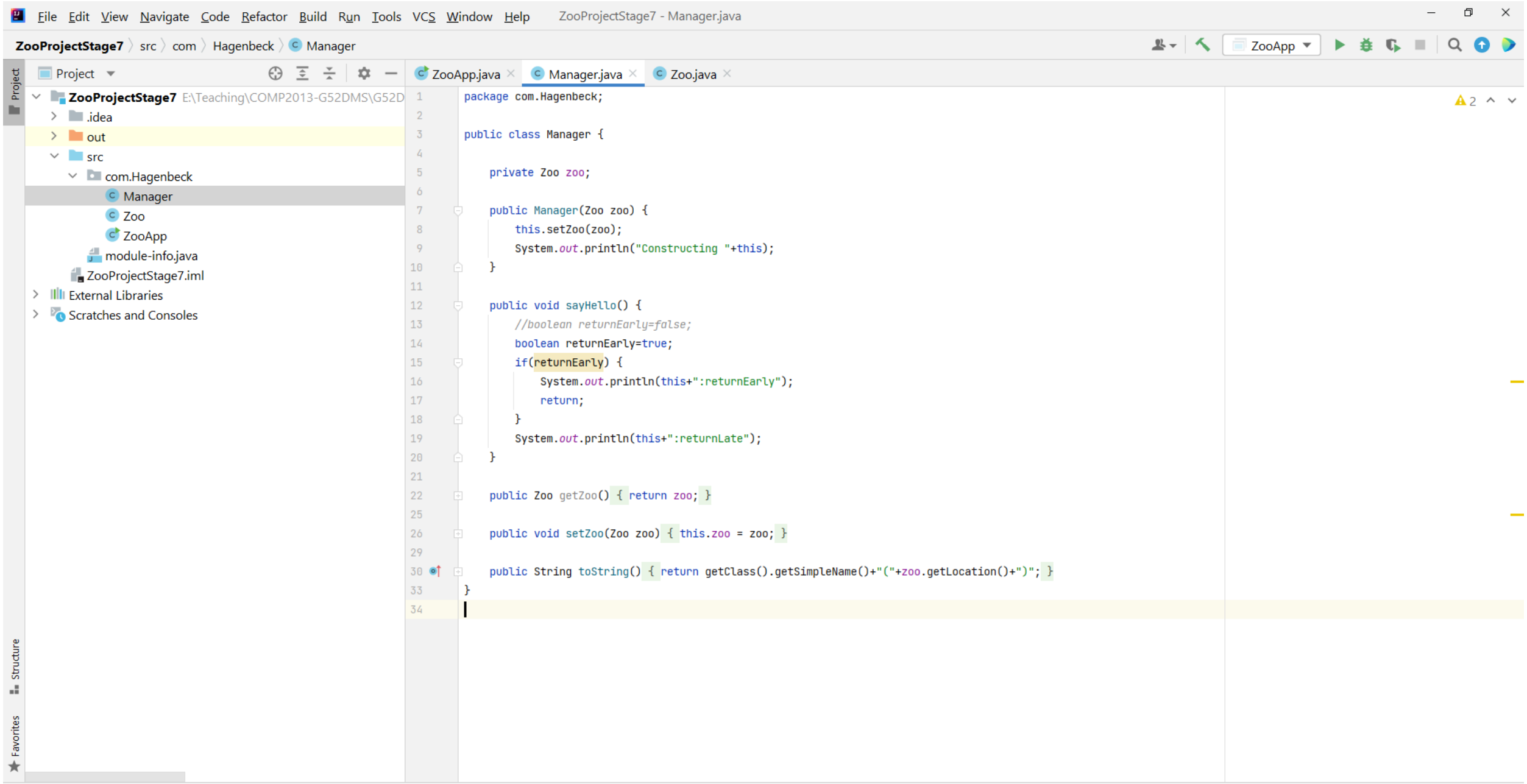
ZooProjectStage7 E:\Teaching\COMP2013-G52DMS\G52D
- .idea
- out
- src
  - com.Hagenbeck
    - Manager
    - Zoo
    - ZooApp
  - module-info.java
- ZooProjectStage7.iml
- External Libraries
- Scratches and Consoles

ZooApp.java | Manager.java | Zoo.java

```
3   public class Zoo {
4
5       private static int numZoosCreated;
6       private String location;
7
8       public Zoo() { this( location: "Unknown"); }
11
12      public Zoo(String location) {
13          numZoosCreated++;
14          //this.location=location;
15          this.setLocation(location);
16          System.out.println("\nConstructing "+this+" No:"+numZoosCreated);
17          doSomething();
18      }
19
20      public void doSomething() {
21          Manager manager=new Manager( zoo: this);
22          manager.sayHello();
23      }
24
25      public String getLocation() { return location; }
28
29      public void setLocation(String location) { this.location = location; }
32      public void changeZoo(Zoo zoo, int avgVisitors) {
33          avgVisitors=200;
34          System.out.println("\n@Local Start:"+zoo);
35          zoo=new Zoo( location: "London");
36          //zoo.setLocation("Munic");
37          //this.setLocation("Amsterdam");
38          System.out.println("\n@Local End:"+zoo);
39      }
40
41      public static int getNumZoosCreated() { return numZoosCreated; }
44      @Override
45      public String toString() { return getClass().getSimpleName()+"("+this.getLocation()+")"; }
48  }
```

TODO Problems Build Terminal Event Log

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago) 3:14 CRLF UTF-8 Tab*


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ZooProjectStage7 - Manager.java
ZooProjectStage7 〉 src 〉 com 〉 Hagenbeck 〉 Manager
ZooApp
Project
ZooProjectStage7 E:\Teaching\COMP2013-G52DMS\G52D
.idea
out
src
com.Hagenbeck
Manager
Zoo
ZooApp
module-info.java
ZooProjectStage7.iml
External Libraries
Scratches and Consoles
ZooApp.java
Manager.java
Zoo.java
package com.Hagenbeck;
public class Manager {
private Zoo zoo;
public Manager(Zoo zoo) {
this.setZoo(zoo);
System.out.println("Constructing "+this);
}
public void sayHello() {
//boolean returnEarly=false;
boolean returnEarly=true;
if(returnEarly) {
System.out.println(this+":returnEarly");
return;
}
System.out.println(this+":returnLate");
}
public Zoo getZoo() { return zoo; }
public void setZoo(Zoo zoo) { this.zoo = zoo; }
public String toString() { return getClass().getSimpleName()+"("+zoo.getLocation()+")"; }
}
TODO
Problems
Build
Terminal
Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK shared indexes // Always download // Download once // Don't show again // Configure... (a minute ago)
34:1
CRLF
UTF-8
Tab*

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help ZooProjectStage7 - Manager.java

ZooProjectStage7 〉 src 〉 com 〉 Hagenbeck 〉 Manager

ZooApp

Project ZooApp.java Manager.java Zoo.java

Run: ZooApp

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\lib\idea_rt.jar=50686:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\bin" -Dfile

Constructing Zoo(Hamburg) No:1
Constructing Manager(Hamburg)
Manager(Hamburg):returnEarly

Constructing Zoo(Munic) No:2
Constructing Manager(Munic)
Manager(Munic):returnEarly

Constructing Zoo(Unknown) No:3
Constructing Manager(Unknown)
Manager(Unknown):returnEarly

Zoo1:Zoo(Munic)
Zoo2:Zoo(Munic)
Zoo3:Zoo(Unknown)

Zoo1:Zoo(Berlin)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)

Constructing Zoo(SanDiego) No:4
Constructing Manager(SanDiego)
Manager(SanDiego):returnEarly

Zoo1:Zoo(SanDiego)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)

@Local Start:Zoo(SanDiego)

Constructing Zoo(London) No:5
Constructing Manager(London)
Manager(London):returnEarly
```

Run TODO Problems Build Terminal Event Log

All files are up-to-date (moments ago) 34:1 CRLF UTF-8 Tab*

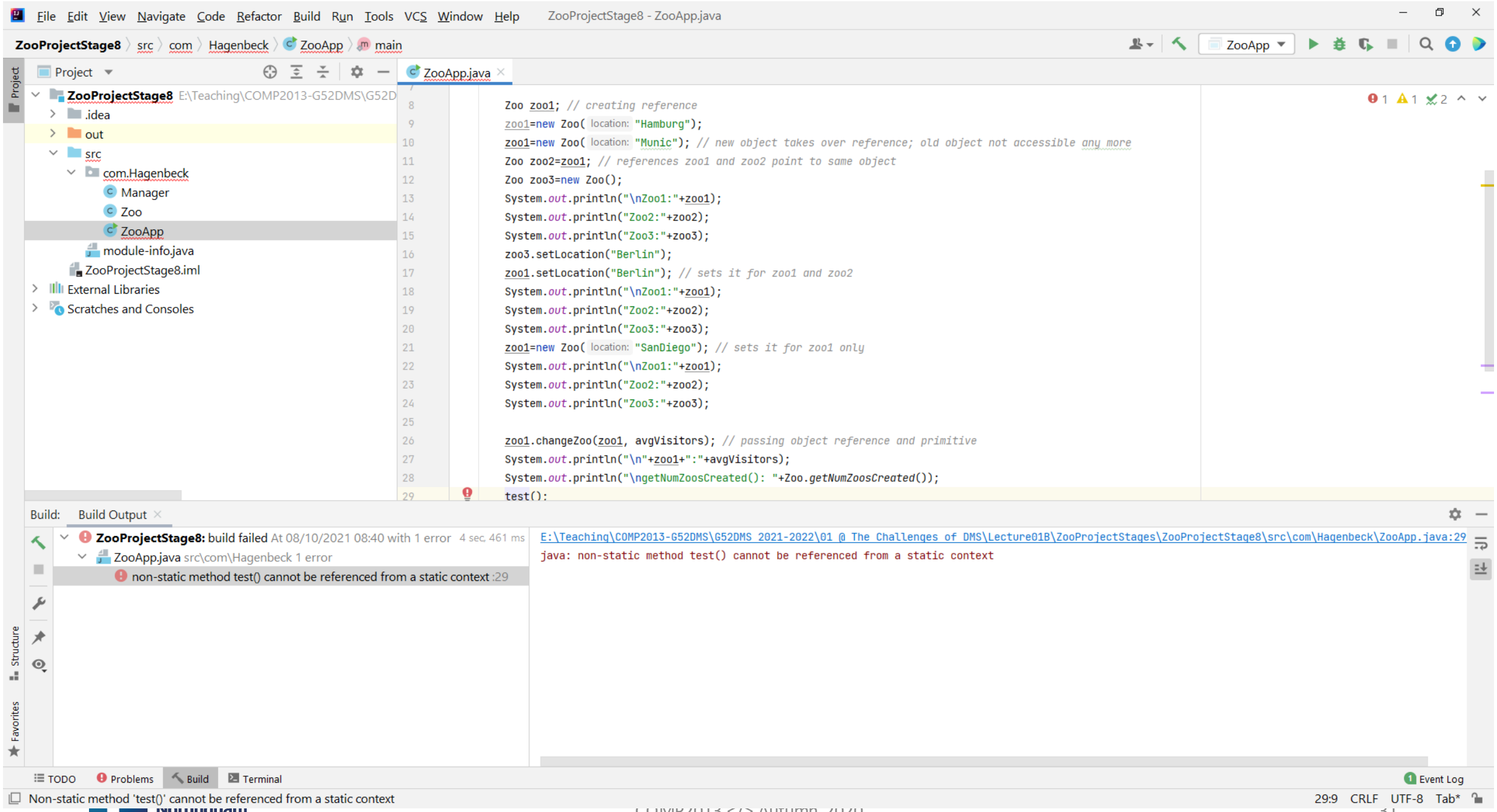
```
Constructing Zoo(Unknown) No:3
Constructing Manager(Unknown)
Manager(Unknown):returnEarly

Zoo1:Zoo(Munic)
Zoo2:Zoo(Munic)
Zoo3:Zoo(Unknown)

Zoo1:Zoo(Berlin)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)

Constructing Zoo(SanDiego) No:4
Constructing Manager(SanDiego)
Manager(SanDiego):returnEarly

Zoo1:Zoo(SanDiego)
Zoo2:Zoo(Berlin)
Zoo3:Zoo(Berlin)

@Local Start:Zoo(SanDiego)

Constructing Zoo(London) No:5
Constructing Manager(London)
Manager(London):returnEarly

@Local End:Zoo(London)

Zoo(SanDiego):100

getNumZoosCreated(): 5

Process finished with exit code 0
```

Static Fields and Methods

- Does this compile?

```
public static void main(String[] args) {
    int avgVisitors=100;

    Zoo zoo1; // creating reference
    zoo1=new Zoo( location: "Hamburg");
    zoo1=new Zoo( location: "Munic"); // new object takes over reference; old object not accessible any more
    Zoo zoo2=zoo1; // references zoo1 and zoo2 point to same object
    Zoo zoo3=new Zoo();
    System.out.println("\nZoo1:"+zoo1);
    System.out.println("Zoo2:"+zoo2);
    System.out.println("Zoo3:"+zoo3);
    zoo3.setLocation("Berlin");
    zoo1.setLocation("Berlin"); // sets it for zoo1 and zoo2
    System.out.println("\nZoo1:"+zoo1);
    System.out.println("Zoo2:"+zoo2);
    System.out.println("Zoo3:"+zoo3);
    zoo1=new Zoo( location: "SanDiego"); // sets it for zoo1 only
    System.out.println("\nZoo1:"+zoo1);
    System.out.println("Zoo2:"+zoo2);
    System.out.println("Zoo3:"+zoo3);

    zoo1.changeZoo(zoo1, avgVisitors); // passing object reference and primitive
    System.out.println("\n"+zoo1+":"+avgVisitors);
    System.out.println("\ngetNumZoosCreated(): "+Zoo.getNumZoosCreated());
    test();
}

public void test() {
    System.out.println("This is a Test");

}
```

ZooProjectStage8


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
ZooProjectStage8 - ZooApp.java
ZooProjectStage8 › src › com › Hagenbeck › ZooApp › main
ZooApp
Project
ZooProjectStage8 E:\Teaching\COMP2013-G52DMS\G52D
.idea
out
src
com.Hagenbeck
Manager
Zoo
ZooApp
module-info.java
ZooProjectStage8.iml
External Libraries
Scratches and Consoles
ZooApp.java
Zoo zoo1; // creating reference
zoo1=new Zoo(location: "Hamburg");
zoo1=new Zoo(location: "Munic"); // new object takes over reference; old object not accessible any more
Zoo zoo2=zoo1; // references zoo1 and zoo2 point to same object
Zoo zoo3=new Zoo();
System.out.println("\nZoo1:"+zoo1);
System.out.println("Zoo2:"+zoo2);
System.out.println("Zoo3:"+zoo3);
zoo3.setLocation("Berlin");
zoo1.setLocation("Berlin"); // sets it for zoo1 and zoo2
System.out.println("\nZoo1:"+zoo1);
System.out.println("Zoo2:"+zoo2);
System.out.println("Zoo3:"+zoo3);
zoo1=new Zoo(location: "SanDiego"); // sets it for zoo1 only
System.out.println("\nZoo1:"+zoo1);
System.out.println("Zoo2:"+zoo2);
System.out.println("Zoo3:"+zoo3);
zoo1.changeZoo(zoo1, avgVisitors); // passing object reference and primitive
System.out.println("\n"+zoo1+":"+avgVisitors);
System.out.println("\ngetNumZoosCreated(): "+Zoo.getNumZoosCreated());
test();
Build: Build Output
ZooProjectStage8: build failed At 08/10/2021 08:40 with 1 error 4 sec, 461 ms
ZooApp.java src\com\Hagenbeck 1 error
non-static method test() cannot be referenced from a static context :29
E:\Teaching\COMP2013-G52DMS\G52DMS 2021-2022\01 @ The Challenges of DMS\Lecture01B\ZooProjectStages\ZooProjectStage8\src\com\Hagenbeck\ZooApp.java:29
java: non-static method test() cannot be referenced from a static context
TODO Problems Build Terminal
Event Log
Non-static method 'test()' cannot be referenced from a static context
29:9 CRLF UTF-8 Tab*

java collections framework

Java Collections Framework

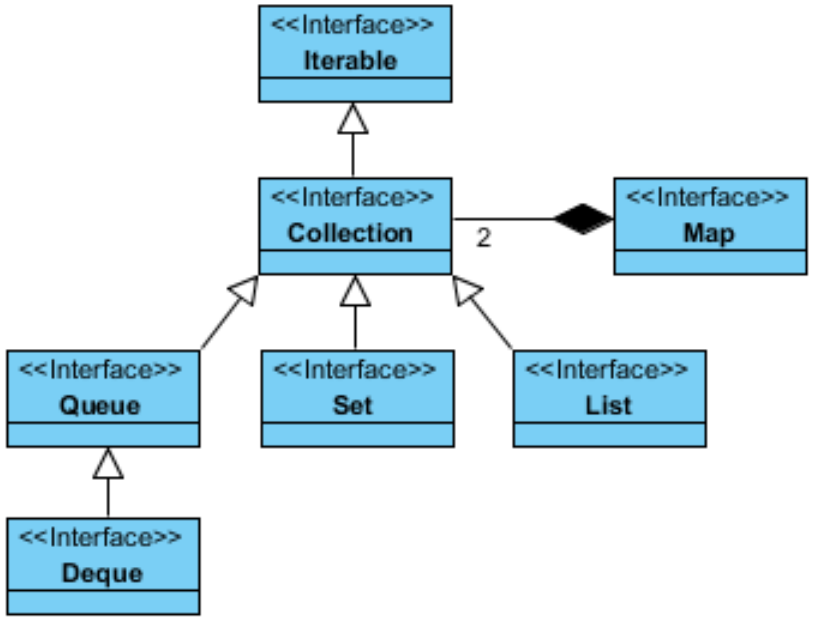
- What do we understand by "Collections" in Java?
  - A collection is an object that represents a group of objects
  - The Collections API is a unified framework for representing and manipulating collections, independent of their implementation
- What does the abbreviation API stand for?
  - Application Programming Interface
- What is the difference between a library and an API?
  - An API is an interface or communication protocol between a client and a server, intended to simplify the building of client-side software. A library contains re-usable chunks of code.

Java Collections Framework

- Java Collections Framework (JCF) principle ideas:
  - We have container objects that contain objects
  - All containers are either "collections" or "maps"
  - All containers provide a common set of method signatures, in addition to their unique set of method signatures
- The Java Collections Framework contains data structures
  - e.g. arrays; lists; maps
- The Java Collections Framework contains algorithmic operations
  - e.g. searching; sorting

Java Collections Framework

- Collection
  - Something that holds a dynamic collection of objects
- Map
  - Defines mapping between keys and objects (two collections)
- Iterable
  - Collections are able to return an iterator object that can scan over the contents of a collection one object at a time



Java Collections Framework

- Core collection framework interfaces
  - **Iterable**: Represents an iterator object
  - **Collection**: Represents a group of objects (elements)
  - **Map**: Maps keys to values; no duplicate keys
  - **Queue**: Represents FIFO queues or LIFO stacks
  - **Deque**: Represents a double ended queue
  - **Set**: A collection that cannot contain duplicate elements
  - **List**: An ordered sequence of elements that allows duplicate elements

- Interface location
  - Most interfaces can be found in the java.util.* package
  - The "Iterable" interface resides in the java.lang.* package

Java Collections Framework

- Classes that implement the collection interfaces typically have names in the form of <Implementation style><Interface>

	Implementation style				
Interface	Hash Table	Resizable Array	Balanced Tree	Linked List	Hash Table + Linked List
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Deque		ArrayDeque		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

- Legacy classes (do not use)
  - Vector (now ArrayList); HashTable (now HashMap); Stack (now ArrayDeque)

For more information refer to: https://nikhilmopidevi.github.io/2017/06/19/Overview-of-the-Java-Collections-Framework/

Doubly-linked list and other data structures are well explained here: https://www.javatpoint.com/doubly-linked-list

Module java.base
Package java.util

Class LinkedList<E>

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.AbstractSequentialList<E>
        java.util.LinkedList<E>

Type Parameters:

E - the type of elements held in this collection

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, Deque<E>, List<E>, Queue<E>

```
public class LinkedList<E>
extends AbstractSequentialList<E>
implements List<E>, Deque<E>, Cloneable, Serializable
```

Doubly-linked list implementation of the `List` and `Deque` interfaces. Implements all optional list operations, and permits all elements (including `null`).

All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index.

Note that this implementation is not synchronized. If multiple threads access a linked list concurrently, and at least one of the threads modifies the list structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more elements; merely setting the value of an element is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the list. If no such object exists, the list should be "wrapped" using the `Collections.synchronizedList` method. This is best done at creation time, to prevent accidental unsynchronized access to the list:

```
List list = Collections.synchronizedList(new LinkedList(...));
```

The iterators returned by this class's `iterator` and `listIterator` methods are *fail-fast*: if the list is structurally modified at any time after the iterator is created, in any way except through the Iterator's own `remove` or `add` methods, the iterator will throw a `ConcurrentModificationException`. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Constructor Summary

"? extends E" means "some type that either is E or a subtype of E"

Constructors

Constructor	
`LinkedList()`	Constructs an empty list.
`LinkedList(Collection<? extends E> c)`	Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`add(int index, E element)`	Inserts the specified element at the specified position in this list.
boolean	`add(E e)`	Appends the specified element to the end of this list.
boolean	`addAll(int index, Collection<? extends E> c)`	Inserts all of the elements in the specified collection into this list, starting at the specified position.
boolean	`addAll(Collection<? extends E> c)`	Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
void	`addFirst(E e)`	Inserts the specified element at the beginning of this list.
void	`addLast(E e)`	Appends the specified element to the end of this list.
void	`clear()`	Removes all of the elements from this list.
Object	`clone()`	Returns a shallow copy of this LinkedList.

add

```
public void add(int index,
                E element)
```

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Specified by:

`add` in interface `List<E>`

Overrides:

`add` in class `AbstractSequentialList<E>`

Parameters:

`index` - index at which the specified element is to be inserted

`element` - element to be inserted

Throws:

`IndexOutOfBoundsException` - if the index is out of range (`index < 0 || index > size()`)

remove

```
public E remove(int index)
```

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Specified by:

`remove` in interface `List<E>`

Overrides:

`remove` in class `AbstractSequentialList<E>`

Parameters:

Java Collections Framework

- Non typesafe collections (do not use)
  - Collection constructors are not able to specify the type of objects the collection is intended to contain
  - Need to cast objects when using them
    - A "ClassCastExeption" will be thrown if we attempt to cast to the wrong type

```
16  public static void main(String[] args) {
17      LinkedList list=new LinkedList();
18      list.add("a string");
19      String s=(String)list.getFirst();
20      System.out.println(s);
21  }
```

```
Run:  Main
"C:\Program Files\Java\jdk-17\bin\
a string

Process finished with exit code 0
```

Java Collections Framework

- Typesafe collections with "Generics"
  - Classes support generics by allowing a type variable to be included in their declaration; type are declared for the reference and constructor

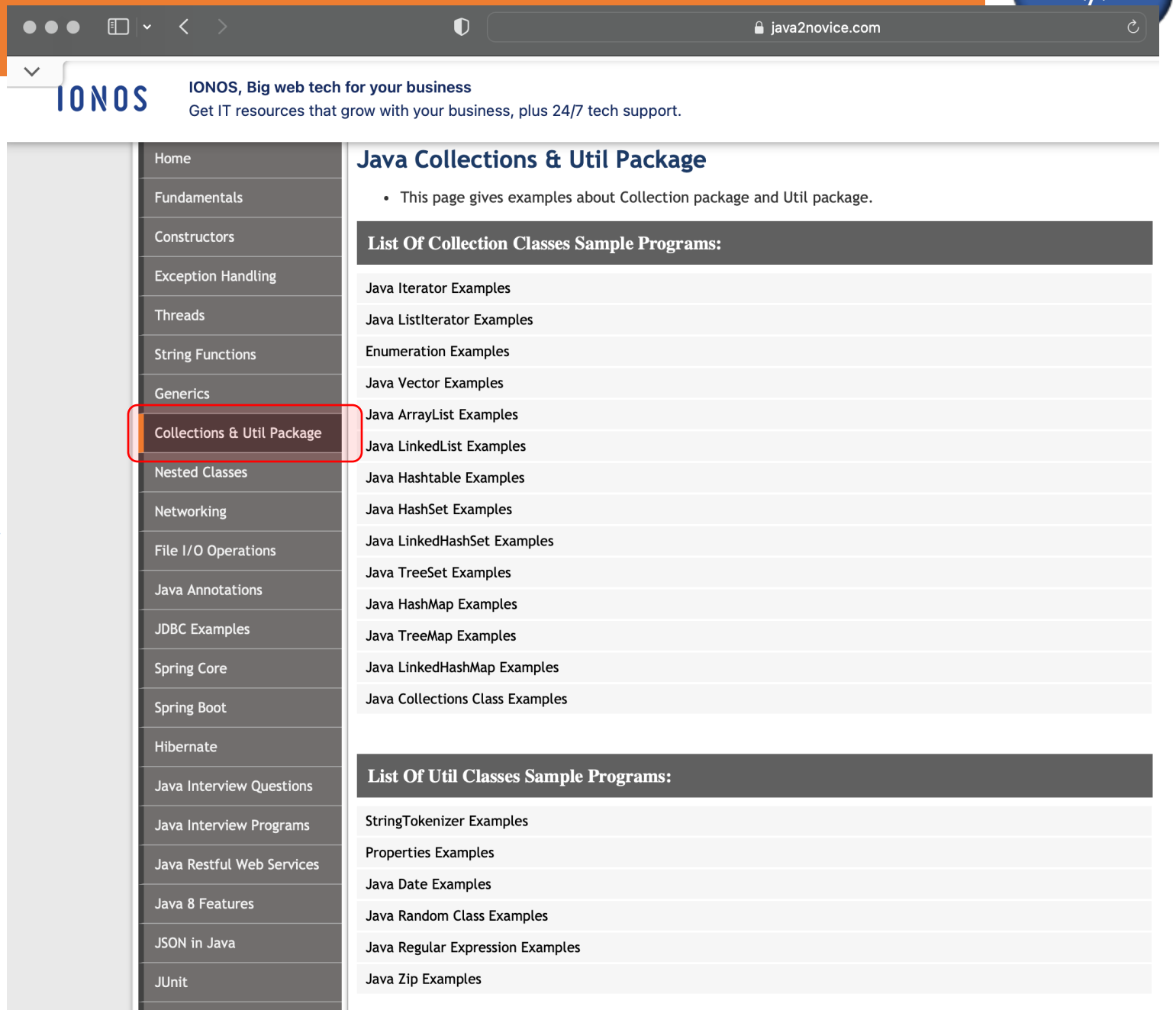
```
35  public static void main(String[] args) {
36      LinkedList<String> list=new LinkedList<>();
37      list.add("a string");
38      String s=list.getFirst();
39      System.out.println(s);
40  }
```

```
Run:  Main
"C:\Program Files\Java\jdk-17\bin\
a string

Process finished with exit code 0
```

- You cannot type a collection using a primitive type (e.g. int)
  - Values of primitive types need to be put into objects of a suitable wrapper class (e.g. Integer) before they can be added to a collection

Useful Resources

http://www.java2novice.com/java-collections-and-util/



Useful Resources

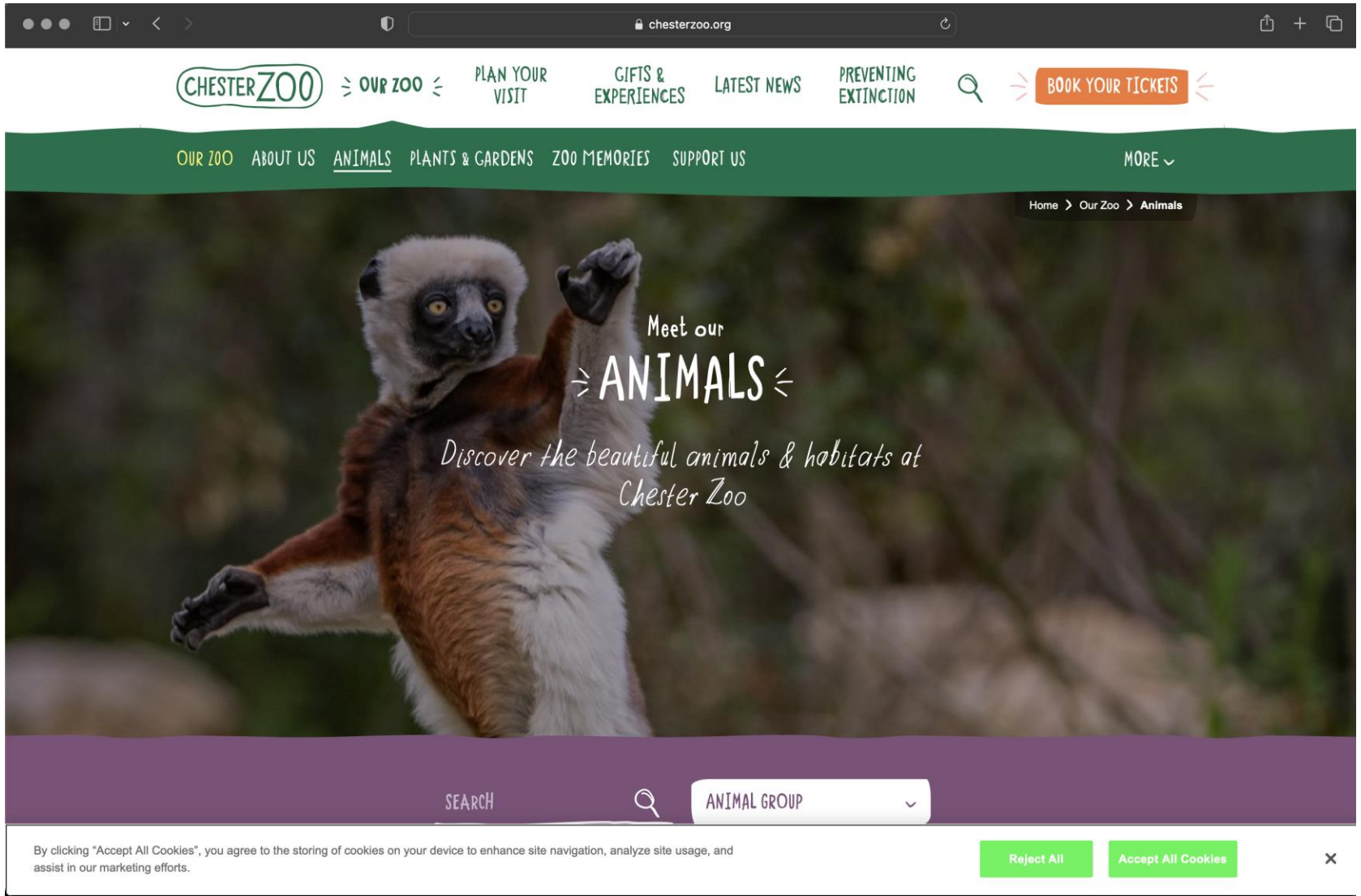
https://www.tutorialspoint.com/java/

implementation of object oriented concepts in java

Concepts we are looking at …

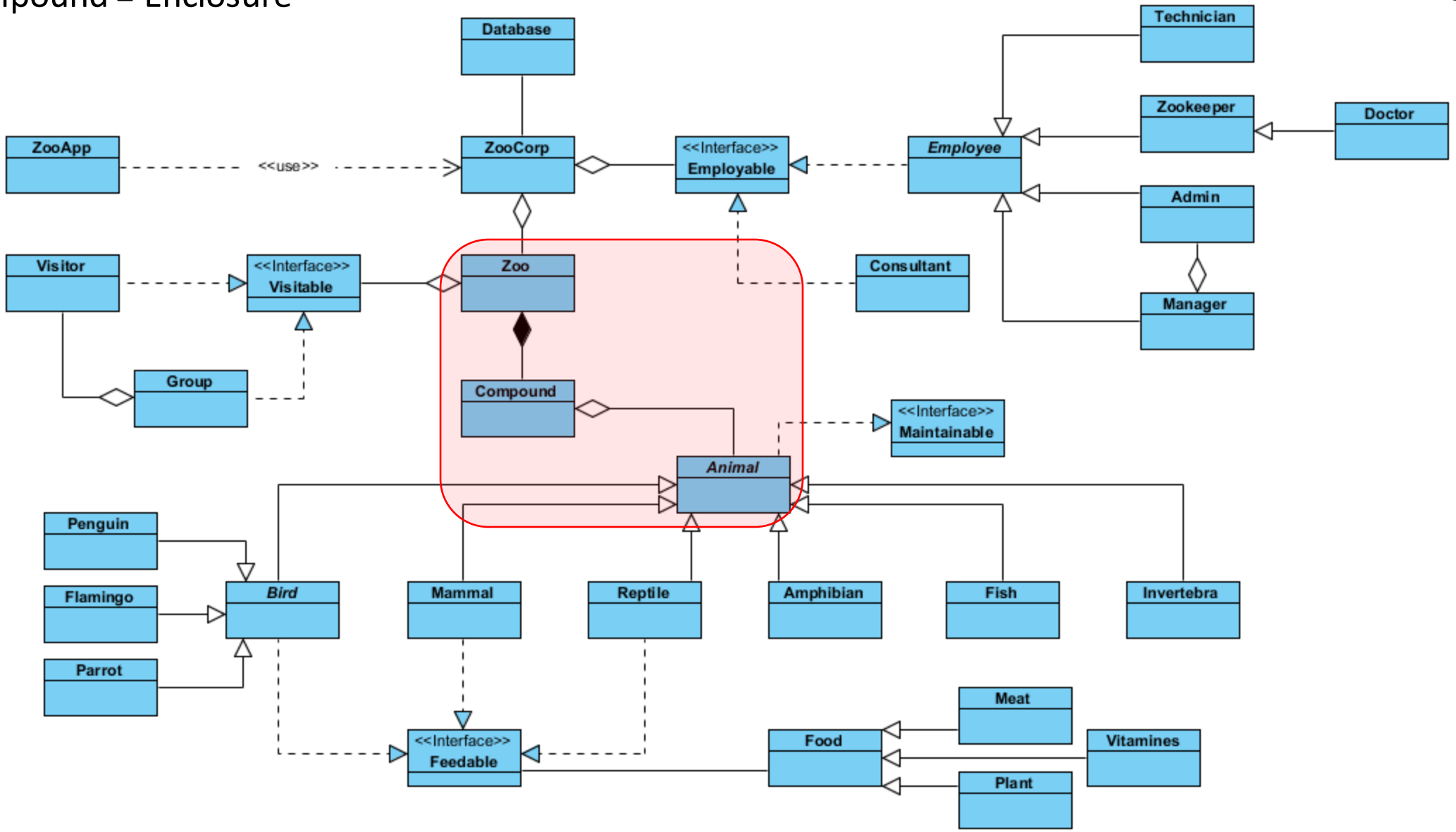
- Aggregation and Composition
- Inheritance and Polymorphism
- Abstract Methods and Classes
- Interfaces

Case Study: Zoo Management



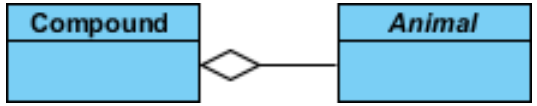
University of Nottingham
UK | CHINA | MALAYSIA

NB: Compound = Enclosure



Aggregation and Composition – 2 min discussion in pairs

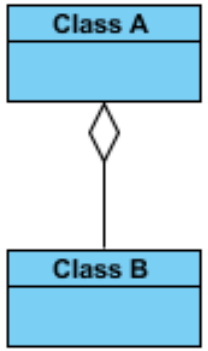
- What is the difference between the Aggregations and Compositions?
  - Aggregation
    - The object exists outside the other, is created outside, so it is passed as an argument



  - Composition
    - The object only exists, or only makes sense inside the other, as a part of the other

Aggregation in Java

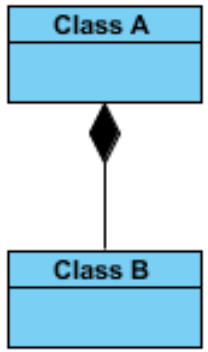
- An object of class B is part of an object of class A (semantically) but the object of class B can be shared and if the object of class A is deleted, the object of class B is not deleted.
  - The object of class A stores the reference to the object of class B for later use
    - Often setter injection is used




```
3  public class A {
4      private B b;
5
6      public void setB(B b){
7          this.b=b;
8      }
9  }
```

Composition in Java

- An object of class A owns an object of class B and the object of class B cannot be shared and if the object of class A is deleted, the object of class B is also deleted
  - The containing object is responsible for the creation and life cycle of the contained object (either directly or indirectly)
    - Composition via member initialisation
    - Composition via constructor initialisation

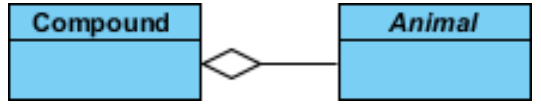



```
3   public class A {
4       private B b=new B();
5   }
```

```
3   public class A {
4       private B b;
5
6       public A(){
7           this.b=new B();
8       }
9   }
```

Aggregation

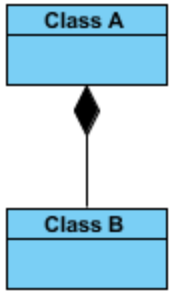
```java
public class Compound {
    private ArrayList<Animal> animals;

    public Compound() { animals=new ArrayList<>(); }
/*
    public void addAnimal() {
        animals.add(new Animal());
    }
*/
    public void addAnimal(Animal animal) { animals.add(animal); }

    public void printInfo() { System.out.println("The compound has "+animals.size()+" animals."); }
}
```




```java
public abstract class Animal {
}
```

Composition

```java
public class Zoo {
    private String location;
    private ArrayList<Compound> compounds;

    public Zoo(String location, int numCompounds) {
        this.setLocation(location);
        this.compounds=new ArrayList<Compound>();
        createCompound(numCompounds);
    }

    public Zoo() { this( location: "Unknown", numCompounds: 1); }

    public void createCompound(int numCompounds) {
        if(numCompounds<1)numCompounds=1;
        for(int i=0;i<numCompounds;i++) {
            this.compounds.add(new Compound());
        }
    }

    public String getLocation() { return location; }

    public void setLocation(String location) { this.location = location; }

    public void printInfo() { System.out.println("The zoo in "+location+" has "+compounds.size()+" compounds."); }
}
```




```java
public class A {
    private B b=new B();
}
```

```java
public class A {
    private B b;

    public A(){
        this.b=new B();
    }
}
```

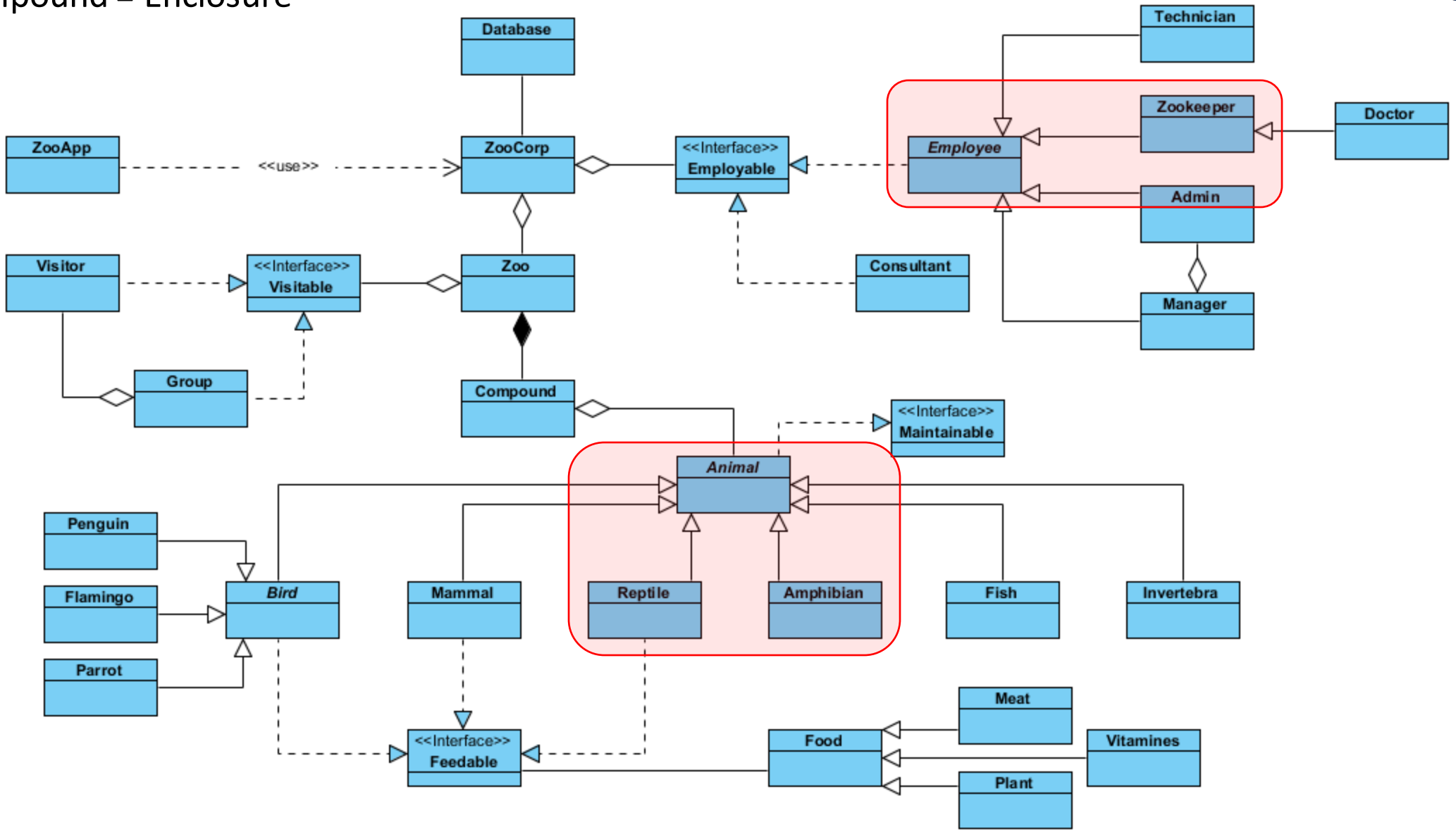
Inheritance

- What is inheritance and why do we use it?
  - Inheritance: Forming new classes based on existing ones
  - Superclass: Parent class being extended
  - Subclass: Child class that inherits behaviour from the parent class
  - "Is-A" relationship

Polymorphism – Job Interview Question, 2 min, pairs

- What is the difference between polymorphism, method overloading, and method overriding?
  - Polymorphism
    - Polymorphism is an object oriented concept
    - Method overloading and method overriding are two forms of polymorphism
  - Method overloading
    - Methods with same the name co-exists in the same class but they must have different method signature (number/type of parameters – think about constructors)
    - Resolved during compile time (static binding)
  - Method overriding
    - Method with the same name is declared in super and sub class (think about the bike1.currentState() example from labsheet 1)
    - Resolved during runtime

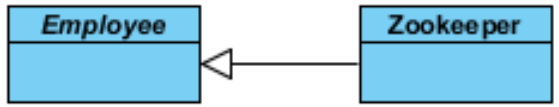
NB: Compound = Enclosure



Inheritance & Polymorphism

```java
public abstract class Employee {
    private String name;
    private double salary;

    public Employee(String name) {
        setName(name);
        setSalary(2000);
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public double getSalary() { return salary; }

    public void setSalary(double salary) { this.salary = salary; }

    public abstract void promotion();
}
```

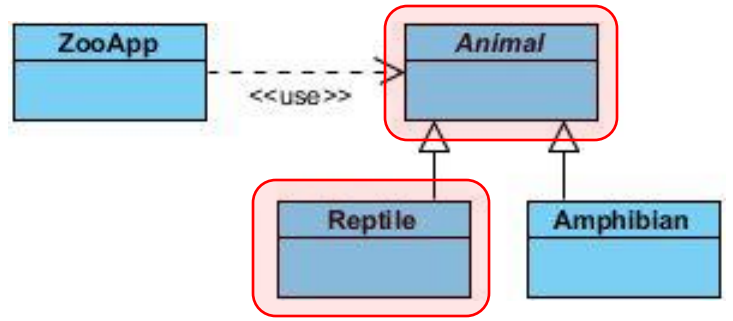
```java
public class Zookeeper extends Employee {

    public Zookeeper(String name) { super(name); }

    @Override
    public double getSalary() {
        double baseSalary=super.getSalary();
        return baseSalary+1000.00;
    }

    @Override
    public void promotion() { super.setSalary(super.getSalary()*1.1); }
}
```



– Notes:
  - A subclass can call its parent's method(s) and constructor(s)
  - A subclass can override its parent's method(s) but not its constructor(s)

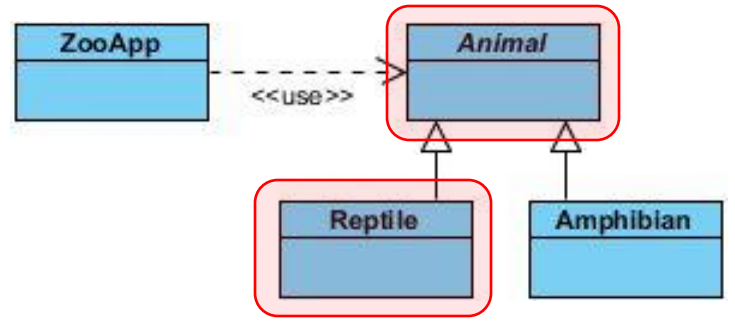
https://stackoverflow.com/questions/5099924/is-constructor-overriding-possible

Inheritance & Polymorphism




```java
public abstract class Animal {
    private String name;

    public Animal(String name) { this.setName(name); }

    public abstract void eat();

    public void enjoy() { System.out.println(this.getClass().getSimpleName()+" enjoys life as animal."); }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
}
```

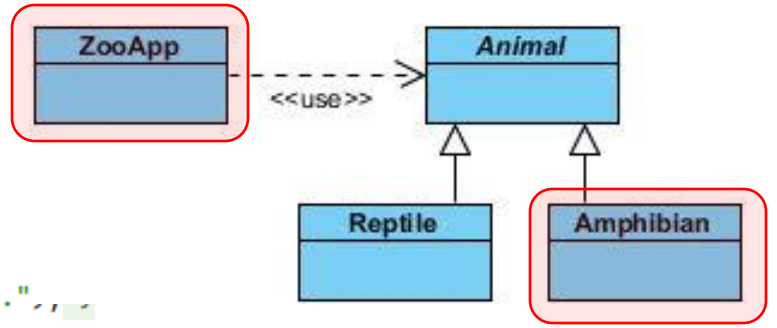
Inheritance & Polymorphism




```java
public class Reptile extends Animal {
    private int numTeeth;

    public Reptile(String name, int numTeeth) {
        super(name);
        this.setNumTeeth(numTeeth);
    }

    @Override
    public void eat() { System.out.println(this.getClass().getSimpleName()+" eats like a reptile."); }

    public int getNumTeeth() { return numTeeth; }

    public void setNumTeeth(int numTeeth) { this.numTeeth = numTeeth; }
}
```

Inheritance & Polymorphism

```java
public class Amphibian extends Animal {

    public Amphibian(String name) { super(name); }

    @Override
    public void eat() { System.out.println(this.getClass().getSimpleName()+" eats like an amphibian."); }

    @Override
    public void enjoy() { System.out.println(this.getClass().getSimpleName()+" enjoys life as amphibian."); }
}
```

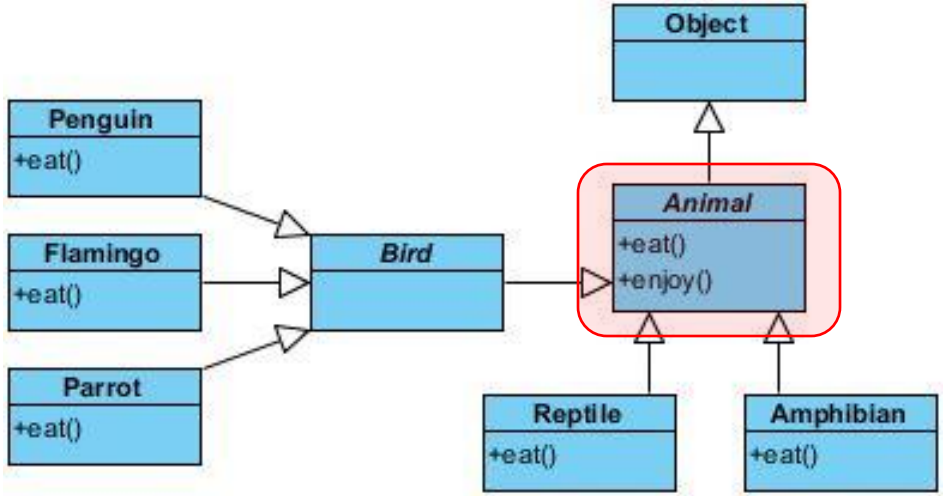


```java
public class ZooApp {

    public static void main(String[] args) {
        Animal animal1=new Amphibian( name: "Frog");
        Animal animal2=new Reptile( name: "Snake", numTeeth: 4);
        Reptile reptile=new Reptile( name: "Turtle", numTeeth: 24);
        animal1.enjoy();
        animal2.enjoy();
        reptile.enjoy();
    }
}
```

```
Run:  Main
"C:\Program Files\Java\jdk-17\bin\ja
Amphibian enjoys life as amphibian.
Reptile enjoys life as animal.
Reptile enjoys life as animal.

Process finished with exit code 0
```

Abstract Methods and Classes

- Animal is an example of an abstract class
  - eat() is an abstract class
  - enjoy() is a concrete class



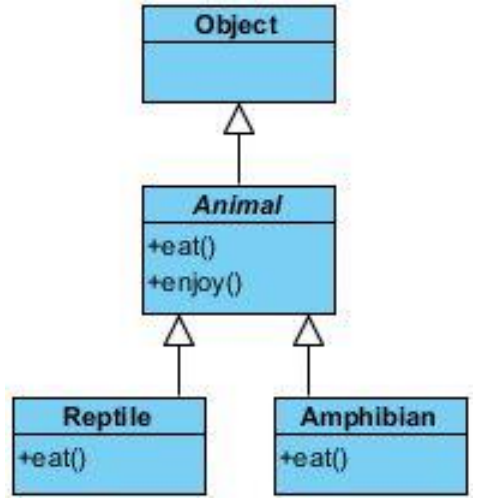
- Any subclass of class Animal has two choices:
  - Define an eat() method
  - Be abstract

- Keep in mind that abstract classes cannot be used to instantiate objects but references to abstract classes are legal (and encouraged)

Abstract Methods and Classes

```java
public class ZooApp {

    public static void main(String[] args) {
        ArrayList<Object> animals=new ArrayList<>();
        Object o=new Reptile( name: "Snake", numTeeth: 4);
        Reptile r=new Reptile( name: "Turtle", numTeeth: 24);
        animals.add(o);
        animals.add(r);
        animals.add(new Amphibian( name: "Frog"));
        while(animals.size()>0) {
            o=animals.remove( index: 0);
            System.out.println(o.toString());
            ((Animal)o).eat();
            ((Animal)o).enjoy();
            System.out.println();
        }
    }
}
```

```
Run:  Main
"C:\Program Files\Java\jdk-17\bin\j
com.zooproject.Reptile@5b6f7412
Reptile eats like a reptile.
Reptile enjoys life as animal.

com.zooproject.Reptile@3941a79c
Reptile eats like a reptile.
Reptile enjoys life as animal.

com.zooproject.Amphibian@506e1b77
Amphibian eats like an amphibian.
Amphibian enjoys life as amphibian.


Process finished with exit code 0
```



We need to cast the Object o into an Animal, in order to access methods eat() and enjoy() as they do not exist in the Object class

Interfaces

- An interface is an abstract type that is used to describe a behavior that classes must implement

- Interfaces may only contain method signature and constant declarations (variable declarations that are declared to be both static and final)
  - Starting with Java 8, default and public static methods can have an implementation in the interface definition
  - Starting with Java 9, private and private static methods can have an implementation in the interface definition

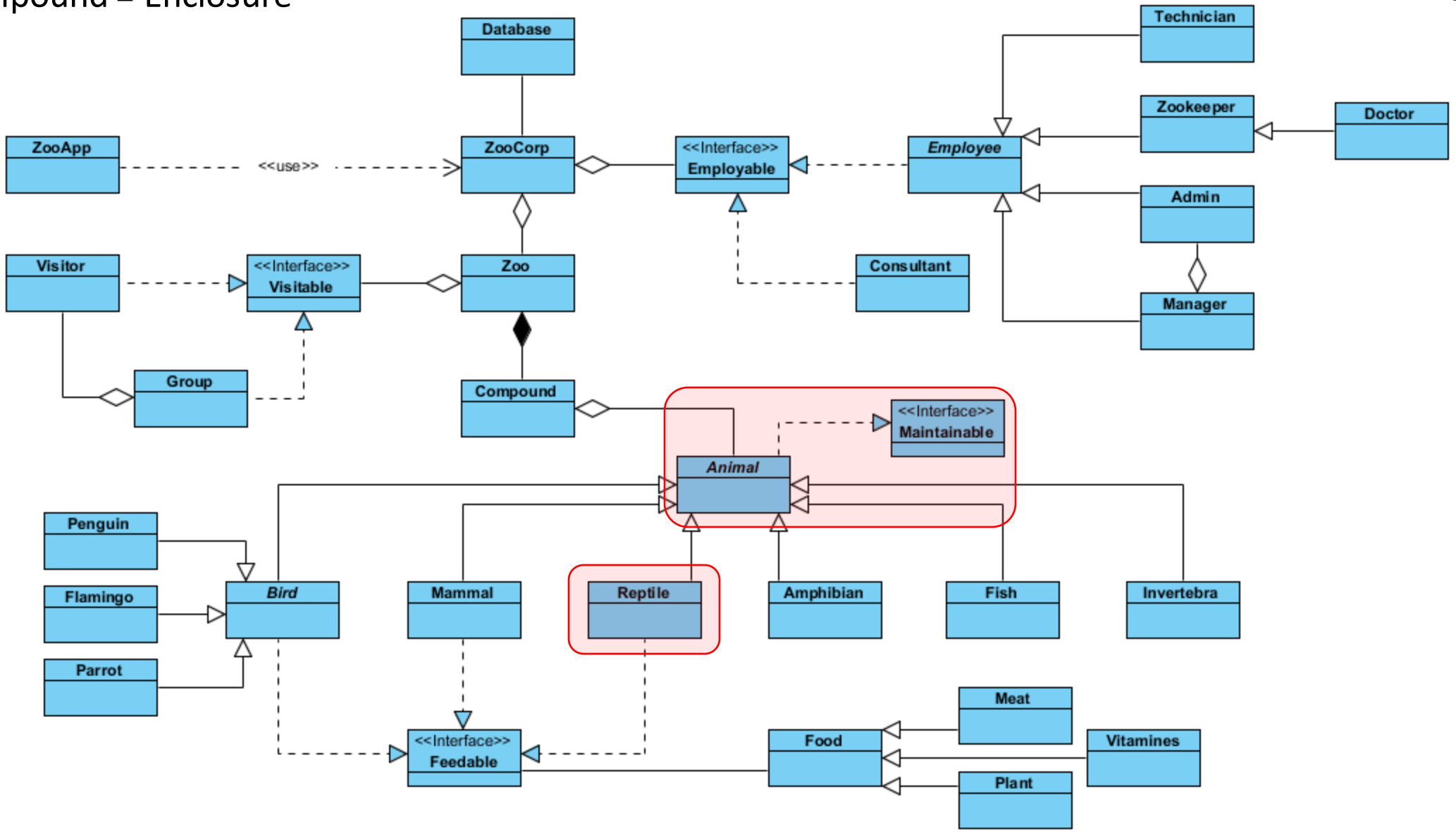
Interfaces

- Interfaces cannot be instantiated, but rather are implemented
- A class that implements an interface must implement all of the non-default methods described in the interface, or be an abstract class
- Interfaces are less restrictive when it comes to inheritance
  - While classes can only ever extend one other class (single inheritance), with interfaces we can choose to implement as many as we like
    - e.g. "public class A extends B implements C, D, E {…}"

Interfaces

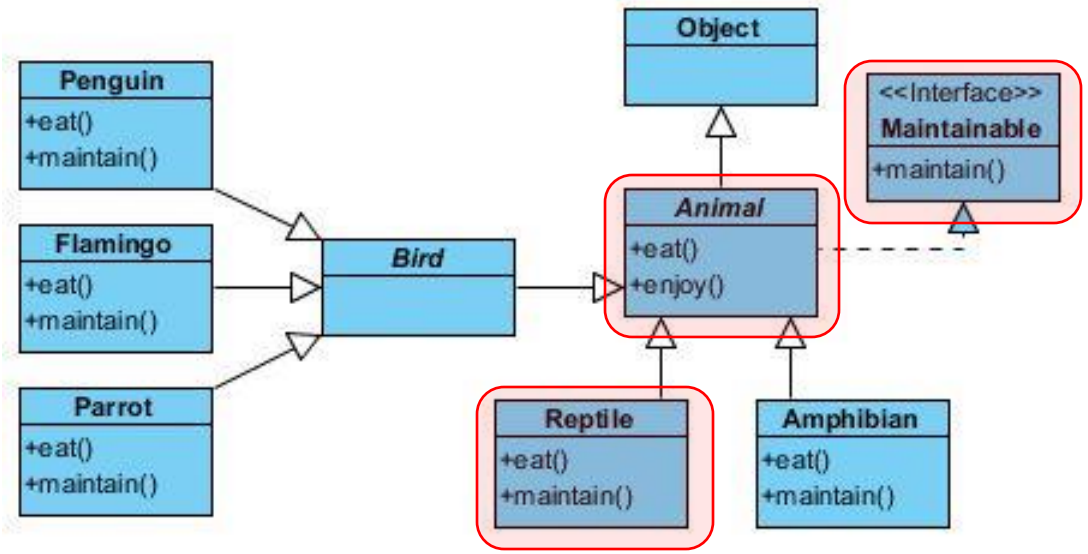
- Some rules:
  - Use the keyword "interface" instead of "class" to declare an interface
  - Implement an interface with the "implements" keyword. A class that implements an interface must provide implementations for all the methods in the interface.
  - Generally all interfaces whose names end with "-able" mark the interface as being able to do something (e.g. Runnable - run(), Executable - execute(), Comparable - compareTo())
  - Similar to classes, you can build up inheritance hierarchies of interfaces by using the "extends" keyword

NB: Compound = Enclosure



Interfaces

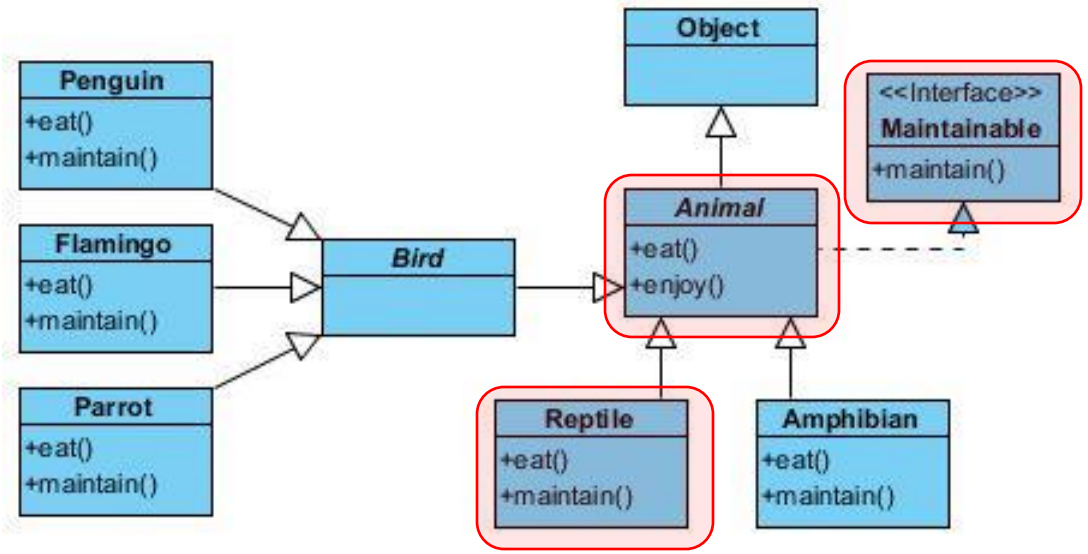
```java
public interface Maintainable {

    public void maintain();
}
```




```java
public abstract class Animal implements Maintainable {
    private String name;

    public Animal(String name) { this.setName(name); }

    public abstract void eat();

    public void enjoy() { System.out.println(this.getClass().getSimpleName()+" enjoys life as animal."); }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
}
```

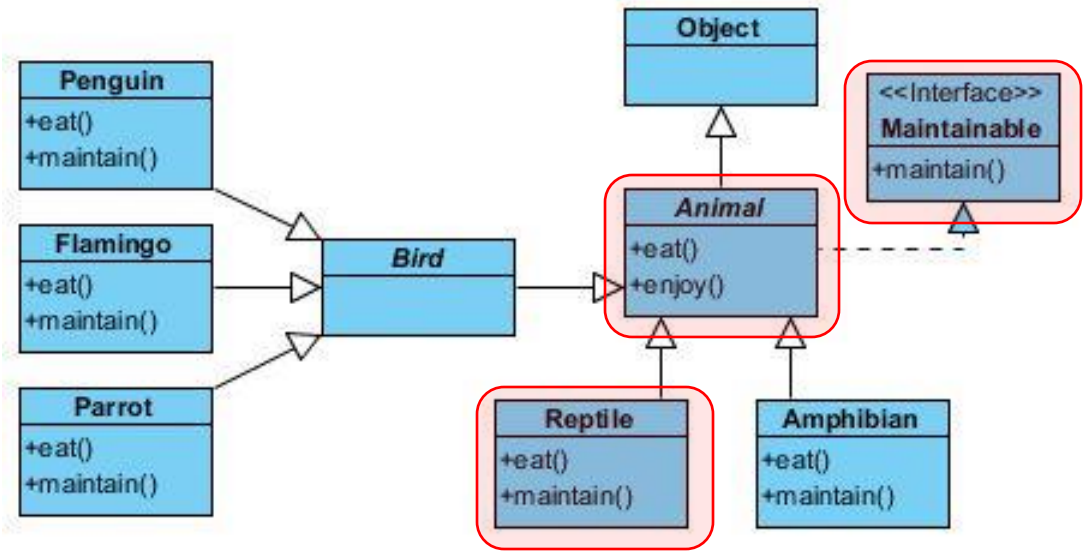
Interfaces

```
3   public class Reptile extends Animal {
4       private int numTeeth;
5
6       public Reptile(String name, int numTeeth) {
7           super(name);
8           this.setNumTeeth(numTeeth);
9       }
10
11      @Override
12      public void eat() { System.out.println(this.getClass().getSimpleName()+" eats like a reptile."); }
15
16      public int getNumTeeth() { return numTeeth; }
19
20      public void setNumTeeth(int numTeeth) { this.numTeeth = numTeeth; }
23
24      @Override
25      public void maintain() { System.out.println(this.getClass().getSimpleName()+" maintains life as reptile."); }
28  }
```



Interfaces

```
Run:  Main
"C:\Program Files\Java\jdk-17\bin\java
com.zooproject.Reptile@27973e9b
Reptile eats like a reptile.
Reptile enjoys life as animal.
Reptile maintains life as reptile.

com.zooproject.Reptile@506e1b77
Reptile eats like a reptile.
Reptile enjoys life as animal.
Reptile maintains life as reptile.

com.zooproject.Amphibian@4fca772d
Amphibian eats like an amphibian.
Amphibian enjoys life as amphibian.
Amphibian maintains life as amphibian.


Process finished with exit code 0
```



Lab Sheet has been released

questions?