Software Development (cs2500)

Lecture 23: Creating Classes from Other Classes

M. R. C. van Dongen

November 15, 2013

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

For Monday

About this Document

Outline

- Today we shall study *inheritance*.
    - With inheritance you can share common code.
    - The common code is written in a common superclass.
    - The common superclass implements common behaviour.
    - Subclasses inherit common behaviour from their superclass.
- We shall carry out two case studies.

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

For Monday

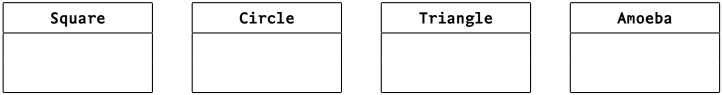
About this Document

Chair Wars Revisited

- Remember Larry and Brad?
- Brad's final solution had five classes:
  - One `Shape` *superclass* for default, common shape behaviour.
  - A dedicated class for each actual shape.
  - Each dedicated class was a *subclass* of the `Shape` class.
  - All, except for `Amoeba`, inherited all behaviour from `Shape`.
  - `Amoeba` *overrode* behaviour for `playSound`, and `rotate`.
  - This let `Amoeba` objects do things differently.
- Larry thought Brad's final class had lots of duplicated code:
  - "Your classes have same code for `playSound` and `rotate`."
  - "This makes it impossible to maintain your code."
  - "For each change, you need to edit 4 classes."
    - Editing *n* class files is *n* times more work than editing 1 file.
    - Each edit increases the probability of errors: more errors.
- But then Brad explained his design.

Introduction
Chair Wars Revisited
Brad Explains
Inheritance
Fota Challenge
For Monday
About this Document

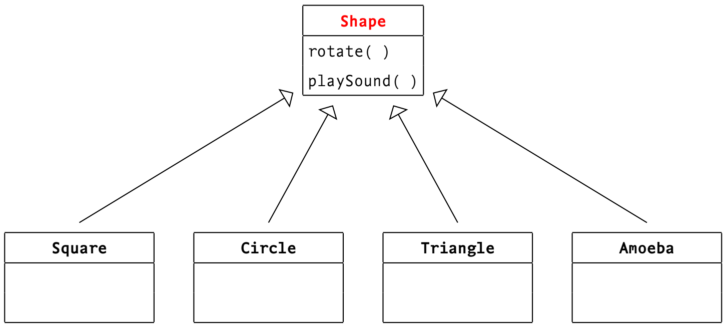
Brad's Final Design



Software Development
M. R. C. van Dongen
Introduction
Chair Wars Revisited
Brad Explains
Inheritance
Fota Challenge
For Monday
About this Document

Brad’s Final Design

Superclass



Introduction

Chair Wars Revisited

Brad Explains

Inheritance

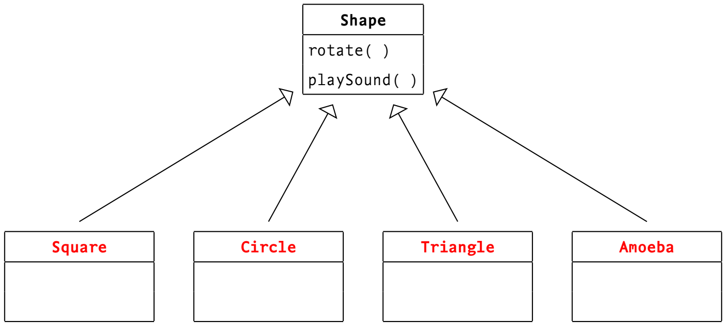
Fota Challenge

For Monday

About this Document

Brad's Final Design

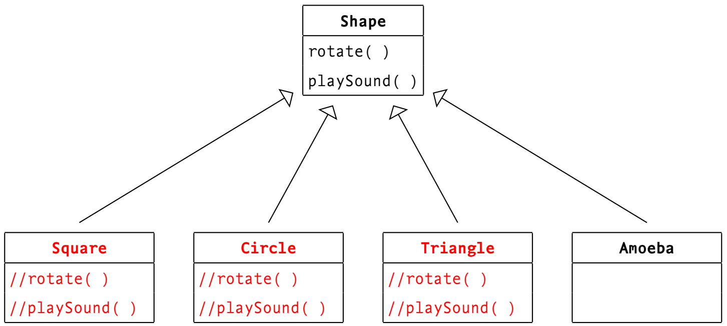
Subclasses



Introduction
Chair Wars Revisited
Brad Explains
Inheritance
Fota Challenge
For Monday
About this Document

Brad's Final Design

Inherit



Introduction

Chair Wars Revisited

Brad Explains

Inheritance

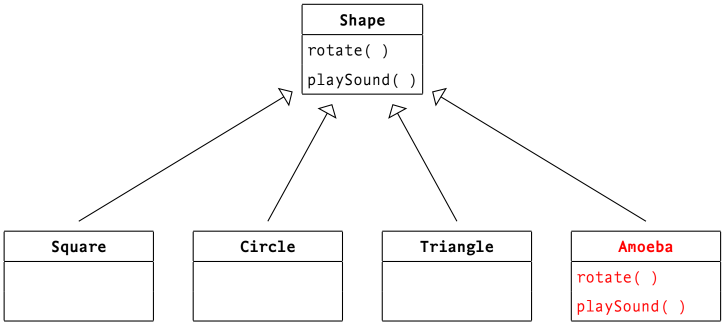
Fota Challenge

For Monday

About this Document

Brad's Final Design

Overrides



Introduction
Chair Wars Revisited
Brad Explains
Inheritance
Fota Challenge
For Monday
About this Document

Inheritance

- There are two main advantages of *inheritance*:
  - Increases ability to reuse implementation effort.
  - Separates class-specific from general code.
- Code is structured in classes so as to maximise reuse.
- *Common* code is put in a common, *more abstract* class.
- The common, more abstract class is called the *superclass*.
- The code in the superclass is shared by *subclasses*.
- The subclasses are more *specific:*
  - Subclass provides same functionality as its superclass.
  - So if the superclass has a method then so does the subclass.
  - Here, the subclass *inherits* the method from its superclass.
  - However, the subclass functionality may be more *specific.*
    - E.g., the subclass may implement a method in a *different* way.
    - Here, the subclass *overrides* the method of its superclass.
  - Subclasses may also have more specific, *additional* behaviour.
- A subclass is said to *extend* its superclass.

Introduction
Chair Wars Revisited
Inheritance
The Class Diagram
Fota Challenge
For Monday
About this Document

Example

- Let's suppose we have a `Surgeon` and a `GP` class.
- Let's also suppose we have a `Doctor` class.
- *Both* `Surgeons` and `GPs` are `Doctors`, they are more specific:
  - A `Surgeon` *is-a* `Doctor`.
  - A `GP` *is-a* `Doctor`.
  - So the `Surgeon` and `GP` classes *extend* the `Doctor` class.
- *Both* have a method called `treatPatient( )`.
  - *Any* `Doctor` has it.
- *Both* have a property `worksAtHospital` (a `boolean`).
  - *Any* `Doctor` has it.
  - For a `Surgeon` it is `true`.
  - For a `GP` it is `false`.
- `Surgeons` and `GPs` differ from `Doctors` *in general*:

Surgeon:	Has additional `makeIncision( )` method.
	Has *special* implementation for `treatPatient`.
	Overrides default `treatPatient( )` implementation.
GP:	Has additional attribute `makesHouseCalls`.
	Has additional method `giveAdvice( )`.

Example Continued
The Common, More General Code

- We put the *more general* code in the `Doctor` class.
- This is the code that *any* `Doctor` should have:
  - `Surgeons` and `GPs` in particular.

Java

```java
public class Doctor {
    public boolean worksAtHospital;

    public void treatPatient( ) {
        // Default patient treatment.
    }




}
```

Example Continued

The Common, More General Code: Did we Forget Anything?

- We put the *more general* code in the `Doctor` class.
- This is the code that *any* `Doctor` should have:
  - `Surgeon`s and `GP`s in particular.

Java

```java
public class Doctor {
    public boolean worksAtHospital;

    public void treatPatient( ) {
        // Default patient treatment.
    }




}
```

Example Continued

The Common, More General Code: Think of Something Any Doctor Does

- We put the *more general* code in the `Doctor` class.
- This is the code that *any* `Doctor` should have:
    - `Surgeon`s and `GP`s in particular.

Java

```java
public class Doctor {
    public boolean worksAtHospital;

    public void treatPatient( ) {
        // Default patient treatment.
    }




}
```

Example Continued

The Common, More General Code: Right!

- We put the *more general* code in the `Doctor` class.
- This is the code that *any* `Doctor` should have:
    - `Surgeon`s and `GP`s in particular.

Java

```
public class Doctor {
    public boolean worksAtHospital;

    public void treatPatient( ) {
        // Default patient treatment.
    }

    public void chargePatient( ) {
        // Let's face it, they all do.
    }
}
```

Example Continued

The More Specific Code: The Surgeon Class

Java

```
public class Surgeon extends Doctor {
    public Surgeon( ) {
        worksAtHospital = true;
    }

    @Override
    public void treatPatient( ) {
        // Specific patient treatment.
    }

    public void makeIncision( ) {
        // Additional behaviour.
    }
}
```

Example Continued

The More Specific Code: The GP Class

Java

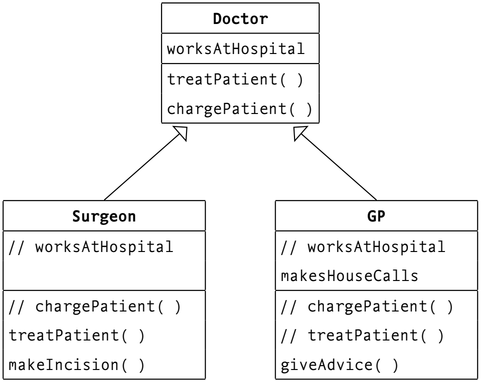
```java
public class GP extends Doctor {
    public boolean makesHouseCalls;

    public GP( boolean makesHouseCalls ) {
        worksAtHospital = false;
        this.makesHouseCalls = makesHouseCalls;
    }

    public void giveAdvice( ) {
        // Additional behaviour.
    }
}
```

Introduction
Chair Wars Revisited
Inheritance
The Class Diagram
Fota Challenge
For Monday
About this Document

The Class Diagram



Introduction

Chair Wars Revisited

Inheritance

The Class Diagram

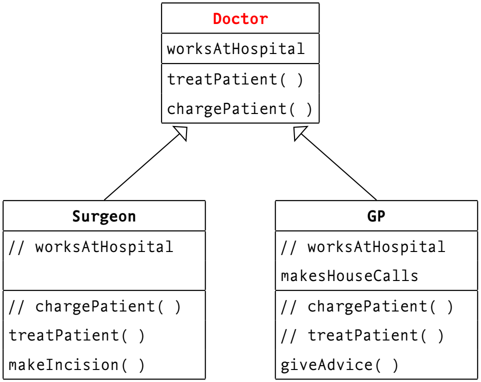
Fota Challenge

For Monday

About this Document

The Class Diagram

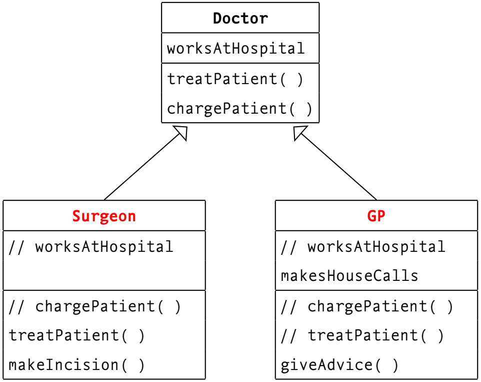
Superclass



Introduction
Chair Wars Revisited
Inheritance
The Class Diagram
Fota Challenge
For Monday
About this Document

The Class Diagram

Subclasses



Introduction

Chair Wars Revisited

Inheritance

The Class Diagram

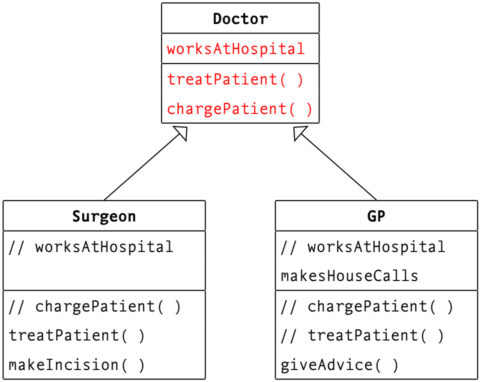
Fota Challenge

For Monday

About this Document

The Class Diagram

Common Methods and Attributes



Introduction

Chair Wars Revisited

Inheritance

The Class Diagram

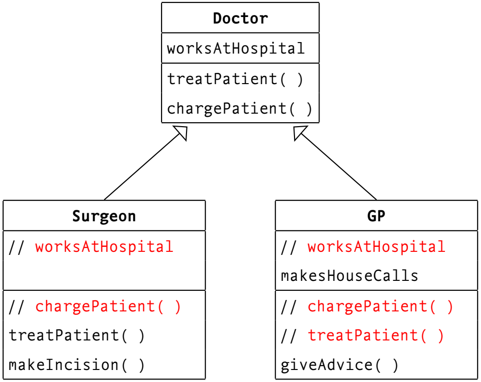
Fota Challenge

For Monday

About this Document

The Class Diagram

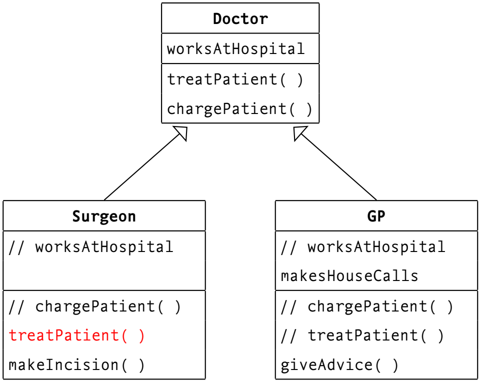
Inherit



Introduction
Chair Wars Revisited
Inheritance
The Class Diagram
Fota Challenge
For Monday
About this Document

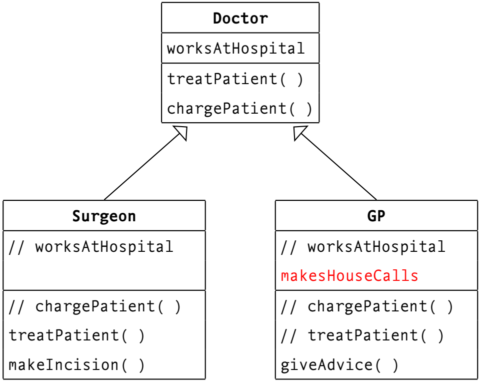
The Class Diagram

Overrides



The Class Diagram

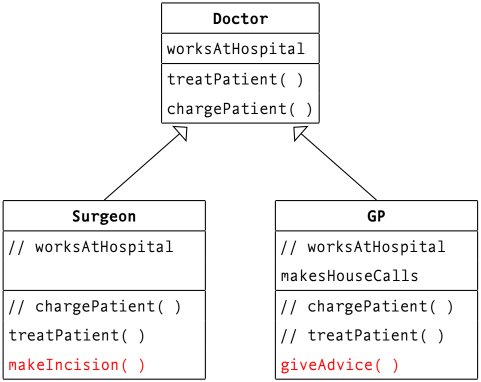
Specific Attribute



Software Development
M. R. C. van Dongen
Introduction
Chair Wars Revisited
Inheritance
The Class Diagram
Fota Challenge
For Monday
About this Document

The Class Diagram

Specific Methods



Introduction
Chair Wars Revisited
Inheritance
The Class Diagram
Fota Challenge
For Monday
About this Document

The Fota Challenge

A Play in Four Acts

Act I:	The Challenge.
Act II:	Larry Presents his Solution.
Act III:	Brad Presents his Solution.
Act IV:	Collecting the prize.

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Act I: The Challenge

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Act I: The Challenge

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

- The Challenge
- Larry's Solution
- Brad's Solution
- The Prize

For Monday

About this Document

Act I: The Challenge

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Act I: The Challenge

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

- The Challenge
- Larry's Solution
- Brad's Solution
- The Prize

For Monday

About this Document

The Spec

- Fota Wildlife Park has lots of animals:
  - A lion;
  - A cat;
  - A wolf;
  - A tiger;
  - A dog; and
  - They’re expecting a hippo.
- Each animal:
  - Has a picture `String`;
  - Has a certain kind of food: grass or meat;
  - Has an integer hunger level;
  - Eats;
  - Makes noise; and
  - Has a roaming behaviour.

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Act I: The Challenge

Oh yeah.

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

- The Challenge
- Larry's Solution
- Brad's Solution
- The Prize

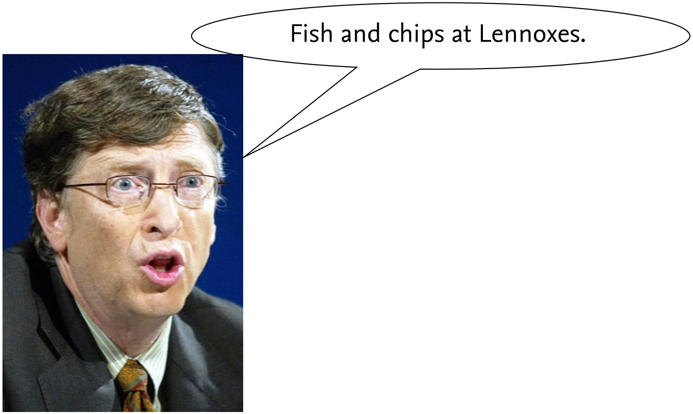
For Monday

About this Document

Act I: The Challenge

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Act I: The Challenge



Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Introducing the Contestants: Meet Larry

- Larry has been taking `Java` lessons with Amy.
- He has just started learning about inheritance.
- He knows inheritance is the key to solving this problem.
- He just knows he will beat Brad.

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Introducing the Contestants: Meet Brad

- Brad is just delighted with this application.
- This is a textbook example of an inheritance application.
- He knows this can't be too difficult.

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile in Larry's Cubicle

- Larry quickly identifies the objects: the animals.
- Following Brad's example, he creates an `Animal` class.
- He puts all the common methods and attributes in this class.

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Meanwhile in Larry's Cubicle

- Larry quickly identifies the objects: the animals.
- Following Brad's example, he creates an `Animal` class.
- He puts all the common methods and attributes in this class.

Java

```
public class Animal {
    public String  picture;
    public boolean eatsGrass;
    public int     hunger;

    ⟨more⟩
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile in Larry’s Cubicle

Java

```
public Animal( String picture,
               boolean eatsGrass,
               int hunger ) {
    this.picture   = picture;
    this.eatsGrass = eatsGrass;
    this.hunger    = hunger;
}

public void eat( ) {           // Default eating behaviour.
    System.out.println( "Eating " + hunger + " portions of " + food( ) + "." );
}
private String food( ) {
    return (eatsGrass ? "grass" : "meat");
}
public void makeNoise( ) { } // Should be overridden.
public void roam( ) { }      // Should be overridden.
public String toString( ) {
    ⟨omitted⟩
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Meanwhile in Larry's Cubicle

Java

```
public class Hippo extends Animal {
    private static final int HIPPO_HUNGER_LEVEL = 10;
    private static final String HIPPO_PICTURE = "hippo.jpg";

    public Hippo( ) {
        picture = HIPPO_PICTURE;
        eatsGrass = true;
        hunger = HIPPO_HUNGER_LEVEL;
    }

    public void roam( ) {
        System.out.println( "I'm Lazy: not roaming." );
    }

    public void makenoise( ) {
        System.out.println( "Grunt." );
    }
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile in Larry's Cubicle

Java

```
import java.util.ArrayList;

public class Main {
    public static void main( String[] args ) {
        ArrayList<Animal> animals = new ArrayList<Animal>( );

        animals.add( new Dog( ) );
        animals.add( new Cat( ) );
        animals.add( new Hippo( ) );
        for (Animal animal : animals) {
            System.out.println( "next: " + animal );
            animal.roam( );
            animal.eat( );
            animal.makeNoise( );
        }
    }
}
```

Larry Presents His Solution

Unix Session

```
$
```

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Larry Presents His Solution

Unix Session

```
$ java Main
```

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge
- The Challenge
- Larry’s Solution
- Brad’s Solution
- The Prize

For Monday

About this Document

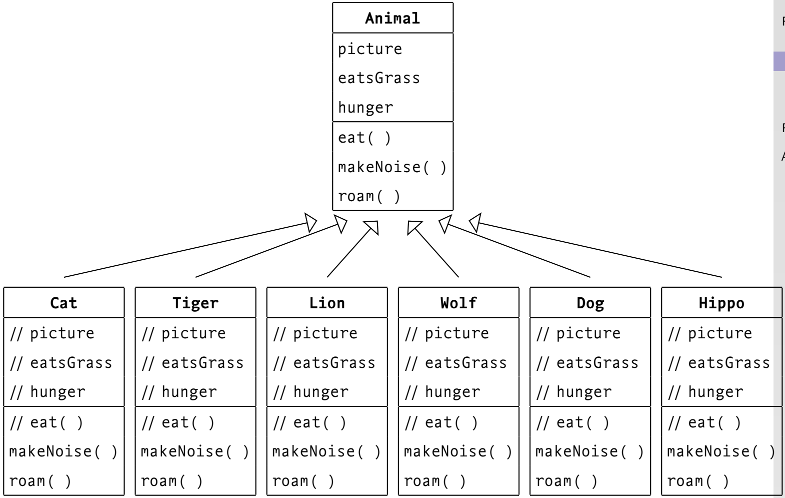
Larry Presents His Solution

Unix Session

```
$ java Main
next: Animal[ picture = dog.jpg, eatsGrass = meat, hunger = 4 ]
Roaming in my pack.
Eating 4 portions of meat.
Arf. Arf.
next: Animal[ picture = cat.jpg, eatsGrass = meat, hunger = 1 ]
Roaming alone.
Eating 1 portions of meat.
Mew. Mew.
next: Animal[ picture = hippo.jpg, eatsGrass = grass, hunger = 10 ]
I'm Lazy: not roaming.
Eating 10 portions of grass.
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Larry's Class Diagram



Introduction

Chair Wars Revisited

Inheritance

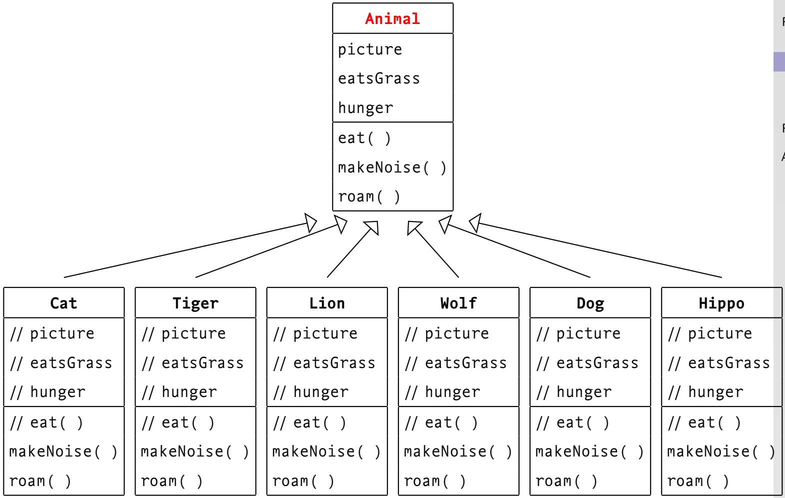
Fota Challenge
- The Challenge
- Larry's Solution
- Brad's Solution
- The Prize

For Monday

About this Document

Larry’s Class Diagram

Superclass



Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry’s Solution

Brad’s Solution

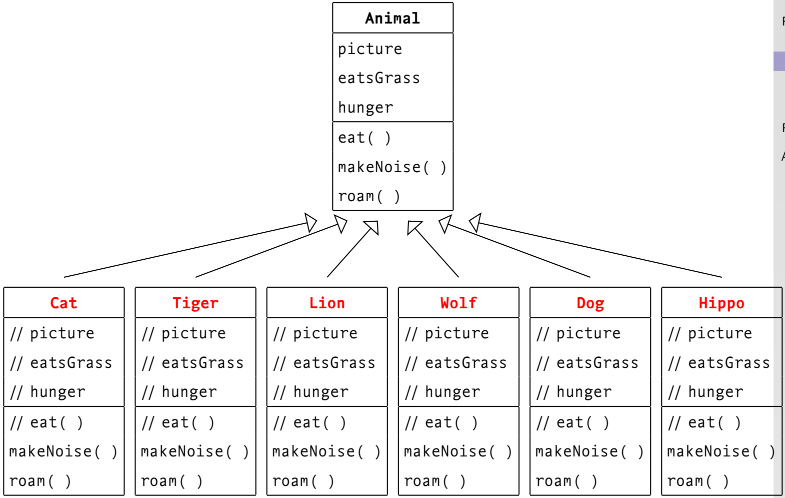
The Prize

For Monday

About this Document

Larry’s Class Diagram

Subclasses



Introduction

Chair Wars Revisited

Inheritance

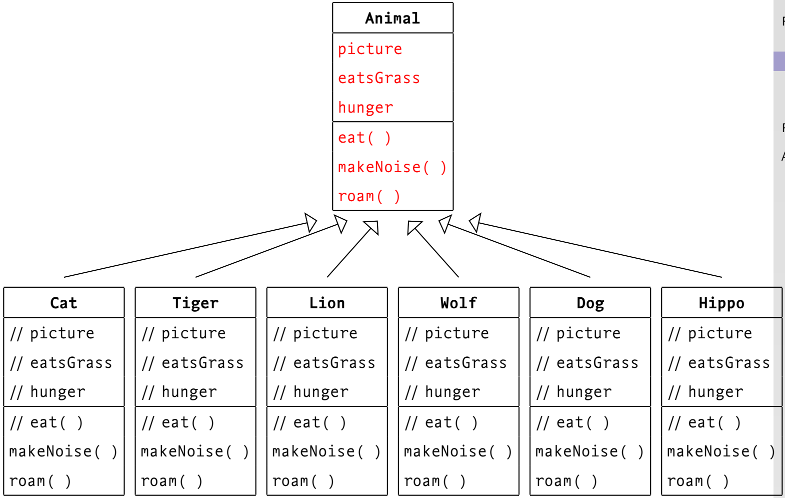
Fota Challenge
- The Challenge
- Larry’s Solution
- Brad’s Solution
- The Prize

For Monday

About this Document

Larry's Class Diagram

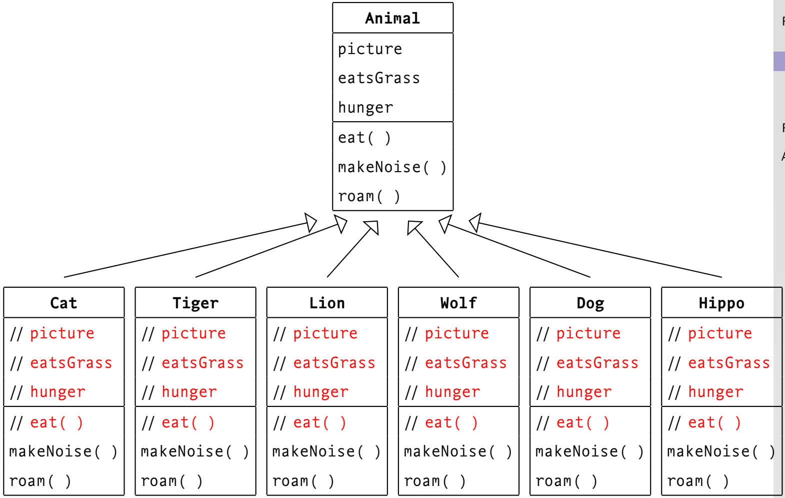
Common Methods and Attributes



Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Larry’s Class Diagram

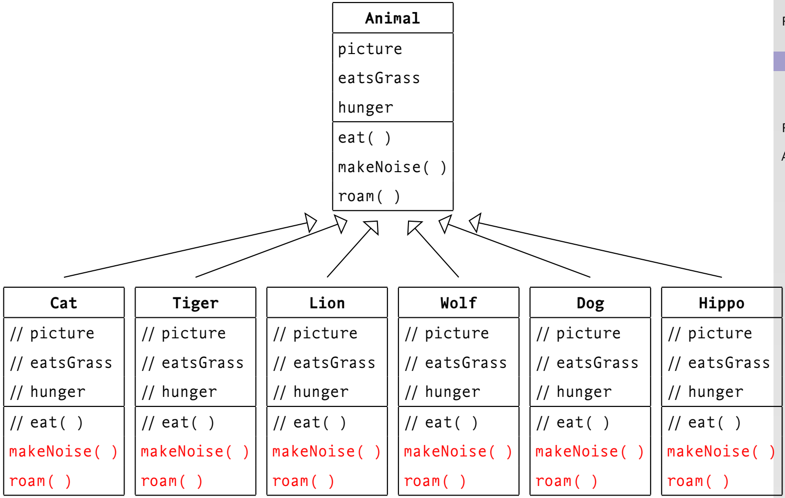
Inherit



Software Development
M. R. C. van Dongen
Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Larry's Class Diagram

Override



Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Did Larry Win?

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Did Larry Win?

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Did Larry Win?

Unix Session

```
next: Animal[ picture = hippo.jpg, eatsGrass = grass, hunger = 10 ]
I’m Lazy: not roaming.
Eating 10 portions of grass.
```

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry’s Solution

Brad’s Solution

The Prize

For Monday

About this Document

Larry Doesn't Get It

- Larry couldn't understand it.
- He had overridden the Hippo's noise method.

Java

```
public void makenoise( ) {
    System.out.println( "Grunt." );
}
```

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Larry Doesn't Get It

- Larry couldn't understand it.
- He had overridden the `Hippo`'s noise method.
- But Amy discovered the error.

Java

```java
public void makenoise( ) {
    System.out.println( "Grunt." );
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Larry Doesn't Get It

- Larry couldn't understand it.
- He had overridden the `Hippo`'s noise method.
- But Amy discovered the error.
- There was a typo in his `Hippo` class.

Java

```
public void makenoise( ) {
    System.out.println( "Grunt." );
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Larry Doesn't Get It

- Larry couldn't understand it.
- He had overridden the `Hippo`'s noise method.
- But Amy discovered the error.
- There was a typo in his `Hippo` class.

Java

```
public void makenoise( ) {
    System.out.println( "Grunt." );
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Larry Doesn't Get It

- Larry couldn't understand it.
- He had overridden the Hippo's noise method.
- But Amy discovered the error.
- There was a typo in his Hippo class.

Java

```
public void makeNoise( ) {
    System.out.println( "Grunt." );
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Larry Doesn’t Get It

- Larry couldn’t understand it.
- He had overridden the `Hippo`’s noise method.
- But Amy discovered the error.
- There was a typo in his `Hippo` class.

Java

```java
@Override // Makes sure we override the right method
public void makeNoise( ) {
    System.out.println( "Grunt." );
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Meanwhile at Brad’s Laptop

- Brad had read about Lennoxes in the *Lonely Planet*.
- Eating there is supposed to be a lifetime experience.
- He is very keen on winning this prize.
- Brad’s design is completely different from Larry’s.

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry’s Solution

Brad’s Solution

The Prize

For Monday

About this Document

Meanwhile at Brad's Laptop

- Brad had read about Lennoxes in the *Lonely Planet.*
- Eating there is supposed to be a lifetime experience.
- He is very keen on winning this prize.
- Brad's design is completely different from Larry's.
- He notices there are really three kinds of animals:

Canines	animals with dog-like behaviour;
Felines	animals with cat-like behaviour; and
Others	animals with other behaviour.

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile at Brad's Laptop

- Brad had read about Lennoxes in the *Lonely Planet.*
- Eating there is supposed to be a lifetime experience.
- He is very keen on winning this prize.
- Brad's design is completely different from Larry's.
- He notices there are really three kinds of animals:

Canines	animals with dog-like behaviour;
Felines	animals with cat-like behaviour; and
Others	animals with other behaviour.

- He decides to build this into his class design.

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile at Brad's Laptop

- Brad creates two additional classes: Canine and Feline.
- Both *extend* the Animal class.

Java

```
public class Canine extends Animal {
    public Canine( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming in my pack." ); }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Meanwhile at Brad’s Laptop

- Brad creates two additional classes: Canine and Feline.
- Both *extend* the Animal class.
- All Canines eat meat.

Java

```
public class Canine extends Animal {
    public Canine( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming in my pack." ); }
}
```

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry’s Solution

Brad’s Solution

The Prize

For Monday

About this Document

Meanwhile at Brad's Laptop

- Brad creates two additional classes: Canine and Feline.
- Both *extend* the Animal class.
- All Canines eat meat.
- All Canines roam in packs.

Java

```
public class Canine extends Animal {
    public Canine( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming in my pack." ); }
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile at Brad's Laptop

- Brad creates two additional classes: `Canine` and `Feline`.
- Both *extend* the `Animal` class.
- All `Canine`s eat meat.
- All `Canine`s roam in packs.

Java

```
public class Canine extends Animal {
    public Canine( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming in my pack." ); }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile at Brad's Laptop

- Brad creates two additional classes: `Canine` and `Feline`.
- Both *extend* the `Animal` class.

Java

```
public class Feline extends Animal {
    public Feline( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming alone." ); }
}
```

Meanwhile at Brad's Laptop

- Brad creates two additional classes: `Canine` and `Feline`.
- Both *extend* the `Animal` class.
- All `Felines` eat meat.

Java

```java
public class Feline extends Animal {
    public Feline( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming alone." ); }
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile at Brad’s Laptop

- Brad creates two additional classes: `Canine` and `Feline`.
- Both *extend* the `Animal` class.
- All `Feline`s eat meat.
- All `Feline`s roam alone.

Java

```
public class Feline extends Animal {
    public Feline( )    { eatsGrass = false; }

    @Override
    public void roam( ) { System.out.println( "Roaming alone." ); }
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Meanwhile at Brad's Laptop

- Brad's design is really clever.
- His design *factors out all common `Canine` behaviour.*
- This simplifies the `Canine` subclasses.
  - All `Canine`s inherit the roaming behaviour.
  - By default, `eatsGrass` is `false` for all `Canine`s.

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Meanwhile at Brad's Laptop

Java

```
public class Dog extends Canine {
    private static final int DOG_HUNGER_LEVEL = 4;
    private static final String DOG_PICTURE = "dog.jpg";
    public Dog( ) {
        picture = DOG_PICTURE;
        // eatsGrass is false by default.
        hunger = DOG_HUNGER_LEVEL;
    }
    // Inherits eating  behaviour from Aninmal class.
    // Inherits roaming behaviour from Canine  class.
    @Override
    public void makeNoise( ) { System.out.println( "Arf. Arf." ); }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Meanwhile at Brad’s Laptop

Java

```
public class Cat extends Feline {
    private static final int CAT_HUNGER_LEVEL = 1;
    private static final String CAT_PICTURE = "cat.jpg";
    public Cat( ) {
        picture = CAT_PICTURE;
        // eatsGrass is false by default.
        hunger = CAT_HUNGER_LEVEL;
    }
    // Inherits eating  behaviour from Aninmal class.
    // Inherits roaming behaviour from Feline  class.
    @Override
    public void makeNoise( ) { System.out.println( "Mew. Mew." ); }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge
- The Challenge
- Larry’s Solution
- Brad’s Solution
- The Prize

For Monday

About this Document

Meanwhile at Brad’s Laptop

Java

```
public class Hippo extends Animal {
    // ⟨constants omitted⟩
    public Hippo( ) {
        picture = HIPPO_PICTURE;
        eatsGrass = true;
        hunger = HIPPO_HUNGER_LEVEL;
    }
    // Inherits eating  behaviour from Aninmal class.
    @Override
    public void roam( ) { System.out.println( "I’m lazy: not roaming." ); }
    @Override
    public void makeNoise( ) { System.out.println( "Grunt." ); }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry’s Solution

Brad’s Solution

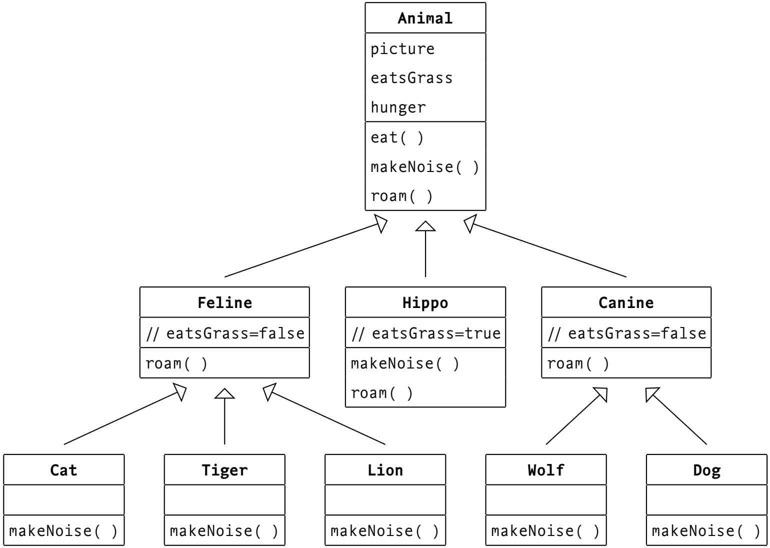
The Prize

For Monday

About this Document

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Brad's Class Diagram



Collecting the Prize

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry’s Solution
Brad’s Solution
The Prize
For Monday
About this Document

Collecting the Prize

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

- The Challenge
- Larry’s Solution
- Brad’s Solution
- The Prize

For Monday

About this Document

Collecting the Prize

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

Collecting the Prize: Cats love Fish

Software Development

M. R. C. van Dongen

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

How Brad Implemented the Canine Class

Poor Style

Java

```
public class Canine extends Animal {
    public Canine( ) {
        eatsGrass = false;
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge
- The Challenge
- Larry's Solution
- Brad's Solution
- The Prize

For Monday

About this Document

How Brad Implemented the Canine Class

Poor Style

Java

```
public class Canine extends Animal {
    public Canine( ) {
        eatsGrass = false;
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

How Brad Implemented the Canine Class

Superclass Implementation Violates Encapsulation

Java

```
public class Canine extends Animal {
    public Canine( ) {
        eatsGrass = false;
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

The Challenge

Larry's Solution

Brad's Solution

The Prize

For Monday

About this Document

How Brad Implemented the Canine Class

Superclass Attributes are Mutable and Cannot be Private

Java

```
public class Canine extends Animal {
    public Canine( ) {
        eatsGrass = false;
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge
- The Challenge
- Larry's Solution
- Brad's Solution
- The Prize

For Monday

About this Document

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Calling the Superclass Constructor

Should be First Call in Constructor

Java

```java
public class Canine extends Animal {
    private static final boolean EATS_GRASS = false;

    public Canine( final String picture, final int hungerLevel ) {
        super( picture, EATS_GRASS, hungerLevel );
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Calling the Superclass Constructor

Superclass Implementation Respects Encapsulation

Java

```
public class Canine extends Animal {
    private static final boolean EATS_GRASS = false;

    public Canine( final String picture, final int hungerLevel ) {
        super( picture, EATS_GRASS, hungerLevel );
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

Calling the Superclass Constructor

Superclass Attributes are *Private* and *Immutable*

Java

```
public class Canine extends Animal {
    private static final boolean EATS_GRASS = false;

    public Canine( final String picture, final int hungerLevel ) {
        super( picture, EATS_GRASS, hungerLevel );
    }

    @Override
    public void roam( ) {
        System.out.println( "Roaming in my pack." );
    }
}
```

Introduction
Chair Wars Revisited
Inheritance
Fota Challenge
The Challenge
Larry's Solution
Brad's Solution
The Prize
For Monday
About this Document

For Monday

For Monday:

- Read the presentation.

Introduction

Chair Wars Revisited

Inheritance

Fota Challenge

For Monday

About this Document

About this Document

- This document was created with `pdflatex`.
- The LaTeX document class is `beamer`.