Week 4: Topics

- Constructors
- Access Modifiers

Constructor in Java

- A constructor in Java is a special type of method

  Used to initialise an object

- Invoked at the time of object creation

  Cannot be called like a normal method

- Default constructor is called automatically when object is created

Constructor Syntax and Rules

```
public class ClassName
{
    ClassName()
    {
    }
}
```

- Constructor name is the same as that of “name of the class”

Does not have a return type

Not even void

Types of Constructors

- There are three types of constructors in Java

  Default constructor

  No-arg constructor

  Parameterized constructor

Default Constructor

- If there is no constructor defined for a class

  Compiler automatically constructs a default constructor

- What is the purpose of a Default Constructor?

  Initializes the objects instance variables to zero/null

No-arg Constructor

- Constructor with no arguments is known as **no-arg constructor**

- Signature is same as default constructor

  However, the body of the constructor can have code

  Unlike a default constructor where the body of the constructor is empty

Problem:

Create a class `Rectangle` with a *no-arg constructor* and print out the `length` and `breadth` of the rectangle.

No-arg Constructor (cont.)

Blackboard: Week4/No-argConstructor/Rectangle.java

Parameterized Constructor

- More often you will need a constructor that accepts one or more parameters
- Why use a *parametrized* constructor?

  Parameterized constructor is used to provide different values to distinct objects

Problem:

Create a class `Rectangle` with a *parameterized constructor* and print out the `length` and `breadth` of the rectangle.

Parameterized Constructor (cont.)

Blackboard: Week4/ParameterizedConstructor/Rectangle.java

A quick tutorial:

Blackboard: Week4/Student.java

Difference Between Constructors & Methods

Java Constructor	Java Method
Used to initialize the state of an object	Used to expose behavior of an object
No return type	Must have a return type
Invoked implicitly	Invoked explicitly
Constructor name must be the same as the class name	Method name may or may not be the same as the class name
Compiler provides a default constructor	Not provided by compiler in any case

Problem:

Write a program to print the names of students by creating a `Student` class.

If no name is passed while creating an object of `Student` class, then the name should be “Unknown”.

Otherwise, the name should be equal to the `String` value passed while creating object of `Student` class.

Blackboard: Week4/Student2

Constructor Overloading

- Overloaded constructors are differentiated on the basis of their

  Type, Number or Order of parameters

- Why is constructor overloading required in Java?

  Sometimes there is a need of initializing an object in different ways

Problem:

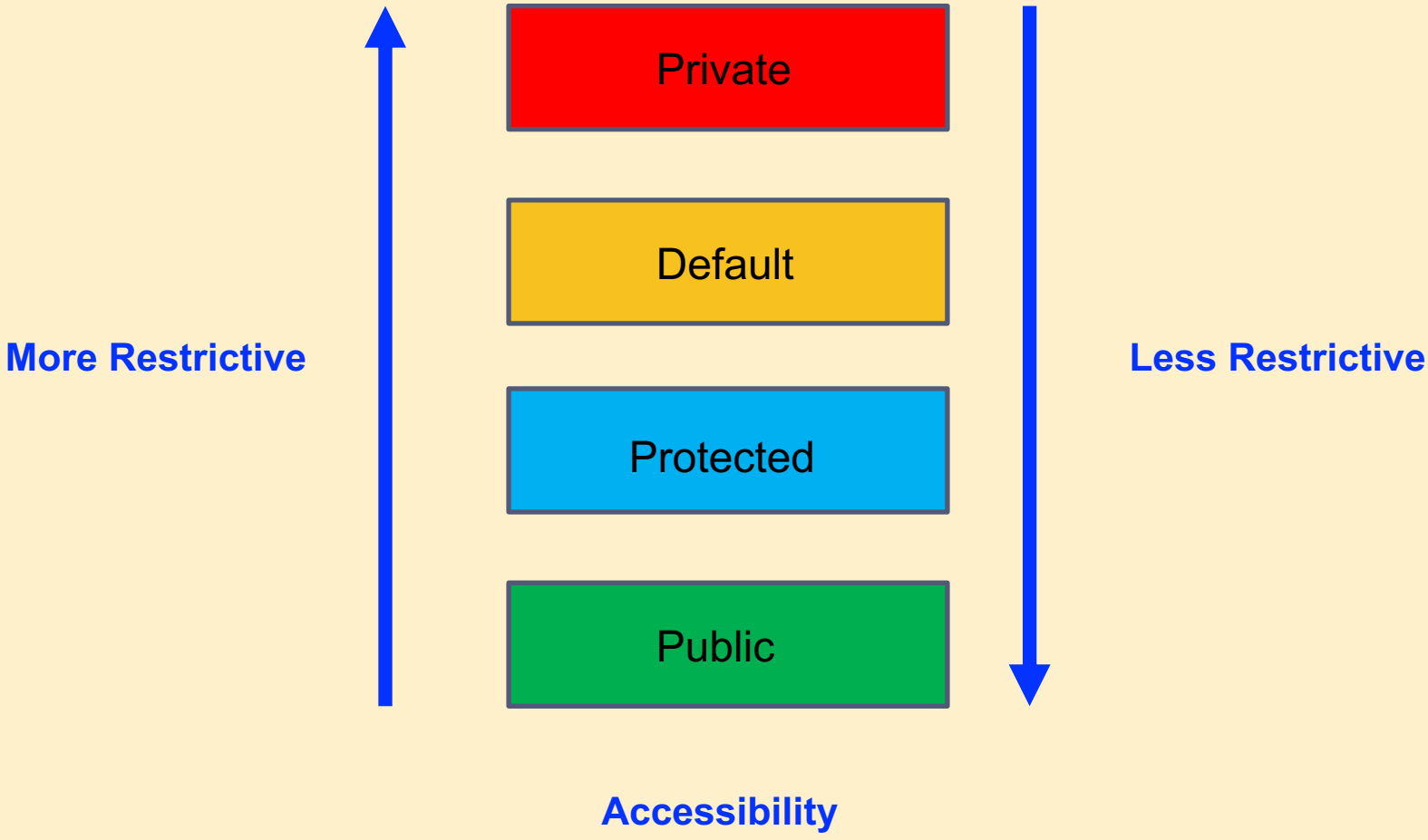
Create a class Box with a default constructor. Overload the constructor such that it takes in all three dimensions as parameters. Define a third variant of the constructor such that the box is a cube. Finally, define a method calculate the volume of the object.

Create three objects - `mybox1`, `mybox2` and `cube` by making use of a different constructor in each instance and print the volume of each object.

Blackboard: Week4/ConstructorOverloading/Box.java

Blackboard: Week4/ConstructorOverloading/Test.java

Access Modifiers in Java



`private` Access Modifier

- The most restrictive access level

  Class and Interfaces cannot be private

- Main mechanism by which an object encapsulates itself
  - Hides data from the outside world

- Methods, variables, and constructors that are declared `private` can only be accessed within the declared class

  Variables that are declared private can be accessed outside the class if public mutator methods are present

`private` Access Modifier

Blackboard: Week4/PrivateAccessModifier

Default Access Modifier

- We do not have to explicitly declare an access modifier for a class, field, method etc.
- A variable or method declared without any access control modifier is available to any other class in the same **package**

  i.e., it is package-private
- A package is a namespace that organizes a set of related classes and interfaces

  Conceptually you can think of packages as being similar to different folders on your computer

Default Access Modifier (cont.)

Blackboard: Week4/DefaultAccessModifier

`protected` Access Modifier

- The `protected` access modifier is accessible within a package and outside the package

  But through inheritance only

- The `protected` access modifier can be applied on a data member, method or constructor

  It cannot be applied on the class

- `Protected` access gives the subclass a chance to use a helper method or variable

  While preventing non-related class from trying to use it

`public` Access Modifier

- Has the widest scope among all other modifiers
  - A class, method, constructor, interface, etc. declared `public` can be accessed from any other class
- However, if the `public` class we are trying to access is in a different package

  Then the class still needs to be imported
- Because of class inheritance

  All public methods and variables of a class are inherited by its subclasses

Access Modifiers Recap

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

A quick tutorial:

`Blackboard: Week4/AccessModifiers`

Which lines are valid and which lines are invalid?