Java™

ORACLE

KIRANJIT
MTECH
ANDROID & IPHONE APPS

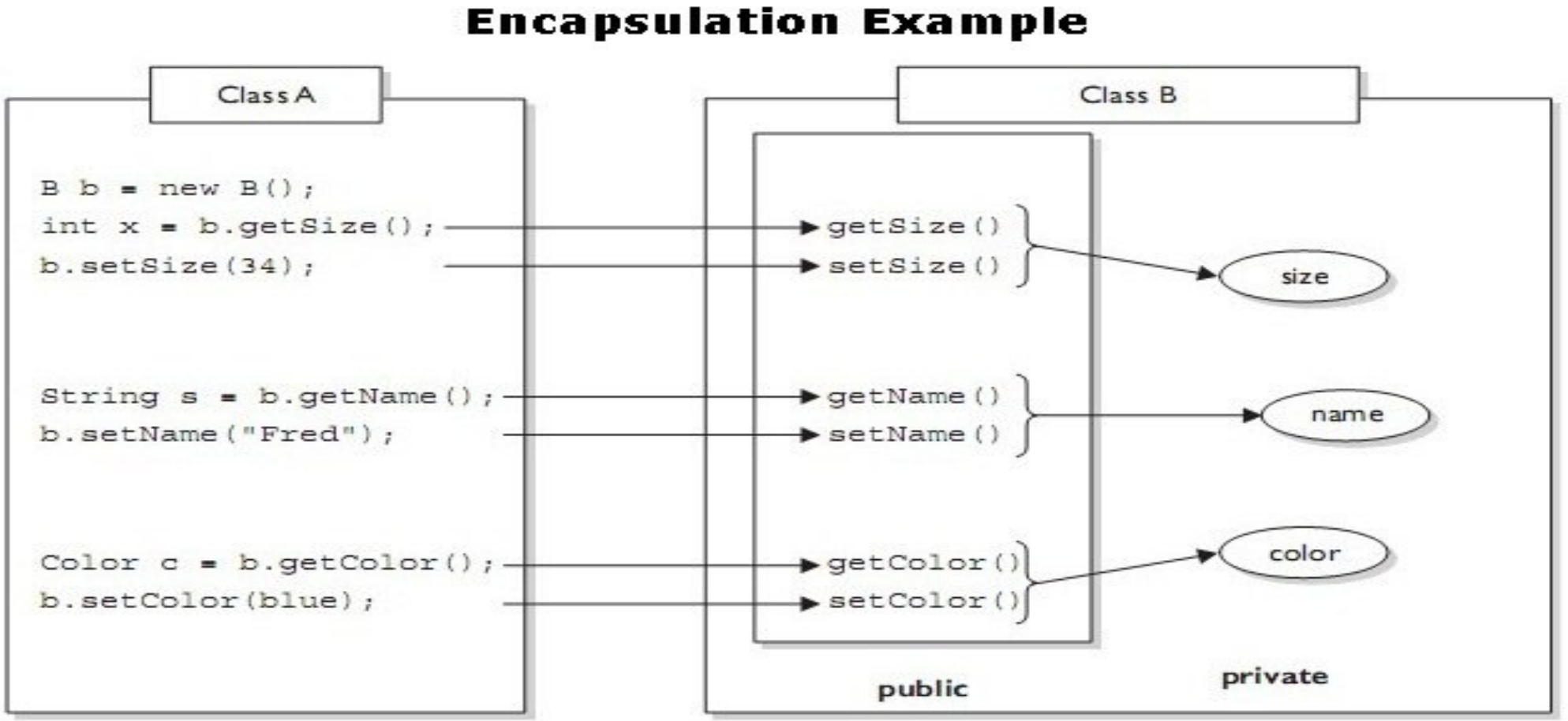
Point of Discussion

- Object Oriented Concepts !

- What is encapsulation in java
- What is inheritance in java
- IS-A and HAS-A relationships
- What is method overriding in java
- What is method overloading in java
- What is constructor in java
- What is coupling and cohesion in java

Point of Discussion

- Encapsulation:- it binds Code + Data it manipulates

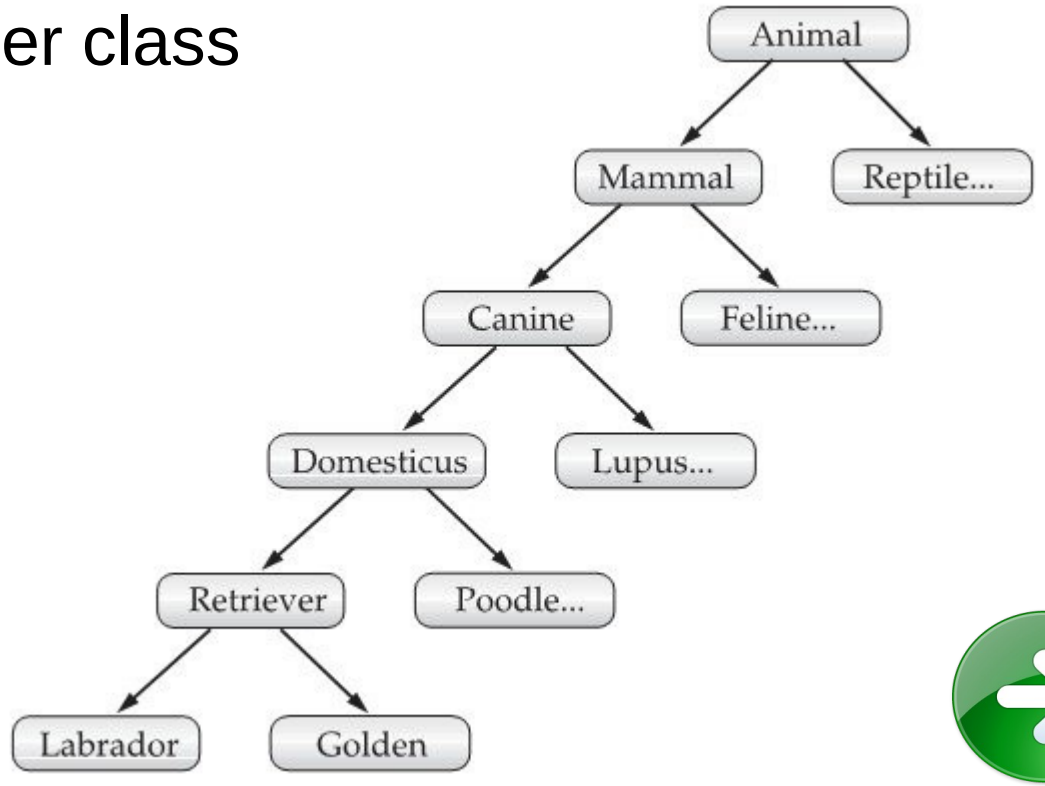


Class A cannot access Class B instance variable data without going through getter and setter methods. Data is marked private; only the accessor methods are public.

Point of Discussion

What is Inheritance?

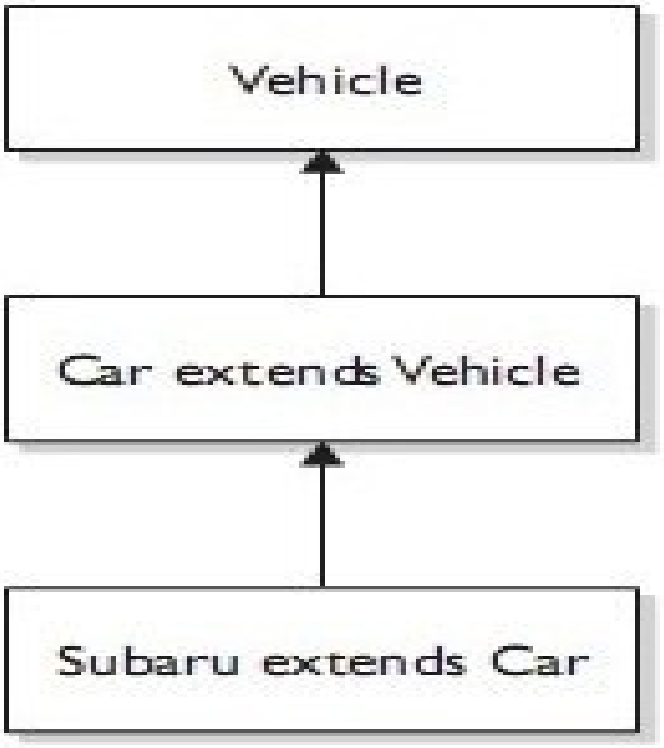
- Inheritance is the process by which one object acquires the properties of another object.
- Why to use ? 1.code resuse 2.to use polymorphism
- What is Sub class & Super class



Point of Discussion

Inheritance promotes IS-A relationship

- Subaru IS-A Car.



Inheritance tree for Vehicle, Car, Subaru

Point of Discussion

PLZ HAVE A LOOK OVER THE PROGRAM

Limitations of Inheritance

- You can only specify one superclass for any subclass that you create.
- Java does not support the inheritance of multiple superclasses into a single subclass.
- You can, as stated, create a hierarchy of inheritance in which a subclass becomes a superclass of another subclass.
- No class can be a superclass of itself.
- Increases the coupling between classes.

Point of Discussion

IS-A RELATIONSHIP	HAS-A RELATIONSHIP
Inheritance is represented by an is-a relationship.	has-a relationship is represented by composition, association or aggregation.
eg. circle is a shape, and Square is a shape.	eg. car is a complex object that has an engine object.
Tightly coupled code.	Loosely coupled code.
Inherits the states and functions of parent class.	One class is using another class by its instance or object
Both the classes are dependent on each other.	Both the classes are independent.

Point of Discussion

METHOD OVERIDING

RULE1:-method must have same signature & return type.

RULE2:-Methods that are not final, private, or static can be overridden.

Importance:-dynamic method dispatch(DMD)

DMD is mechanism by which a call to a overiden method is resolved during run time rather than compile time.

DMD is called Run time Polymorphism.

(SEE D PRGORAM AND HAVE A CONCLUSION.....)

Conclusion of DMD

- if a superclass contains a method that is overridden by a subclass, then when different types of objects are referred to through a superclass reference variable, different versions of the method are executed.

? oops..

Point of Discussion

Overloading.....

- "Multiple methods with same name but different signatures".
- Compile Time Polymorphism.

Example:

```
public int sum(int a, int b) {
        return (a + b);}
// valid method overloading
public float sum(float a, float b) {
        return (a + b);}

// Invalid method overloading : change in only return type
public double sum(float a, float b) {
        // Duplicate method sum(float, float)
        return (a + b); }
```

Point of Discussion

WHAT ABOUT CONSTRUCTORS?

Constructor has the same name as the class in which it resides and is syntactically similar to a method.

KEYPOINTS:

1.The constructor name must match the name of the class.

2.Constructors can use any access modifier, including private.

3.Constructors must not have a return type.

4.The default constructor is always a no-arg constructor.

5.Interfaces do not have constructors. Interfaces are not part of an object's inheritance tree.

6.can be overloaded.

```
class Horse {
        Horse() { } // constructor
        void doStuff() {
        Horse(); // calling the constructor - illegal!
              }
        }
```

Point of Discussion

- What is parameterized or overloaded constructor in java.

```
class Foo {
  int size;
  String name;
  // parameterized constructor
  Foo(String name, int size) {
    this.name = name;
    this.size = size;
  }
```

// That means the following will fail to compile:

Foo f = new Foo(); // won't compile, no matching constructor

// but the following will compile:

Foo f = new Foo("Fred", 43); // No problem. Arguments match with the Foo constructor.

Point of Discussion

Dont worry .....

Lastly.....

Coupling

Coupling is the degree to which one class knows about another class.

// Tightly coupled class design - Bad thing

```
class DoTaxes {
    float rate;
    float doColorado() {
        SalesTaxRates str = new SalesTaxRates();
        rate = str.salesRate; // ouch
        // this should be a method call:
        // rate = str.getSalesRate("CO");
        // do stuff with rate
    }
}
```

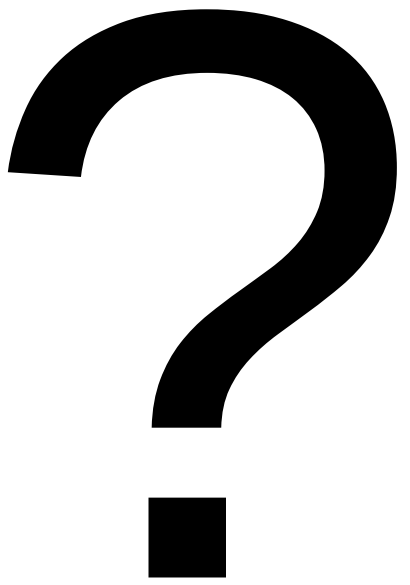
```
class SalesTaxRates {
    public float salesRate; // should be private
    public float adjustedSalesRate; // should be private
    public float getSalesRate(String region) {
        salesRate = new DoTaxes().doColorado(); //
ouch again!
        // do region-based calculations
        return adjustedSalesRate;
    }
}
```

cohesion

```
class BudgetReport {
    Options getReportingOptions() {
    }
    void generateBudgetReport(Options o) {
    }
}
class ConnectToRDBMS {
    DBconnection getRDBMS() {
    }
}
class PrintStuff {
    PrintOptions getPrintOptions() {
    }
}
class FileSaver {
    SaveOptions getFileSaveOptions() {
    }
}
```

Instead of one class that does everything, we have broken the system into four main classes.

NO QUESTION PLZZZZ...



Now its my turn

- Can we overload a constructor ?

- Is multiple inheritance supported in java ?

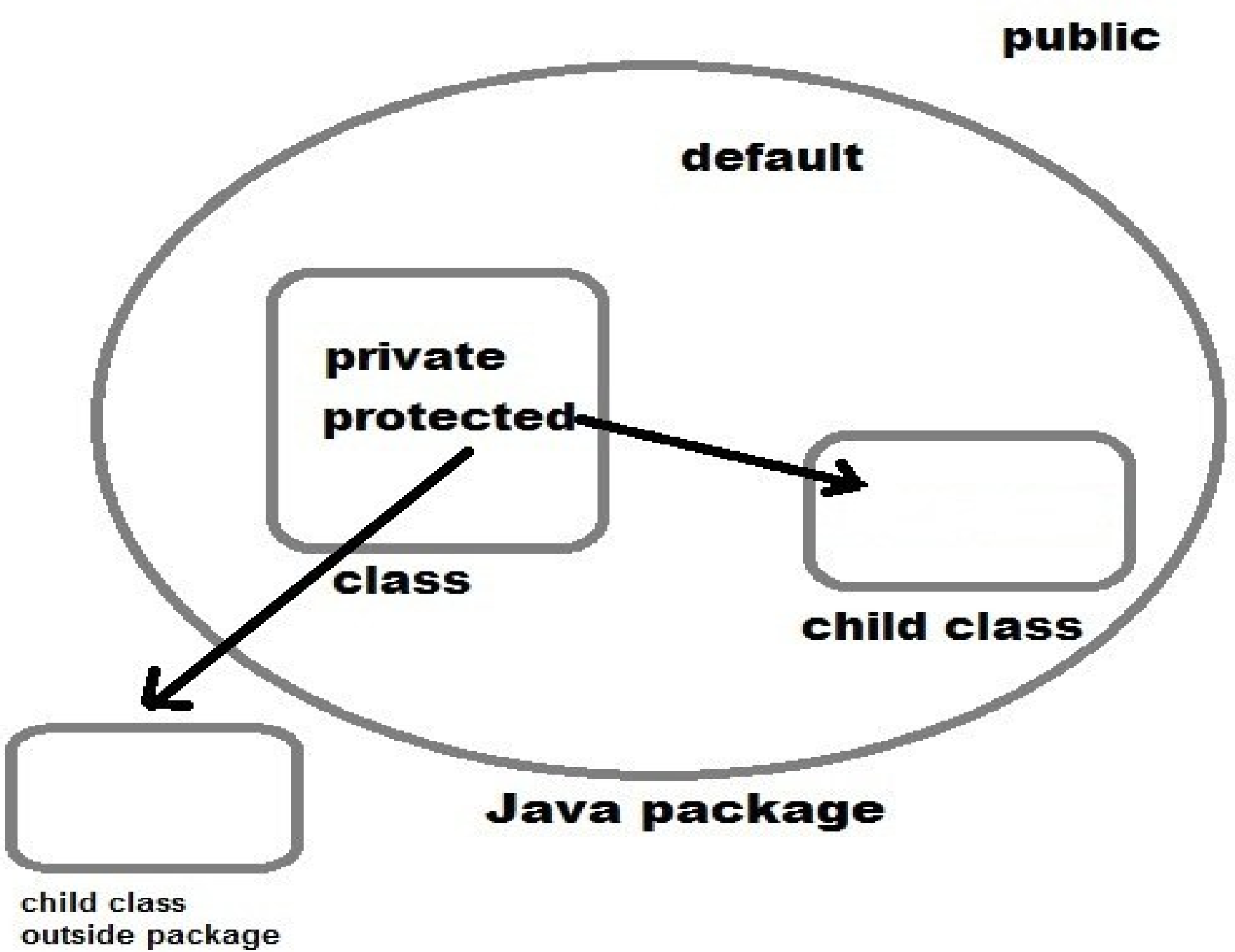
- Dynamic method despatch is _____ time Polymorphism.

?

OVERLOADING is a _____ time Polyorphism.

- Which approach is better ?

  Cohesion or Coupling


public
default
private
protected
class
child class
Java package
child class
outside package


THANK YOU