Java Parents to Child Class Inheritance and Polymorphism

2019 Lecture 2

Preview on inheritance

- The way a class inherit the allowed behaviour and allowed attribute of the parent class
- Use key word extends
- public / private / protected controls rules of inheritance

Inheritance

- A way that a child class can share some of the parent class behaviour
- Both Attribute and Method can be a inheritance to child class
- Child class can override parent class behaviour

Inheritance Rule

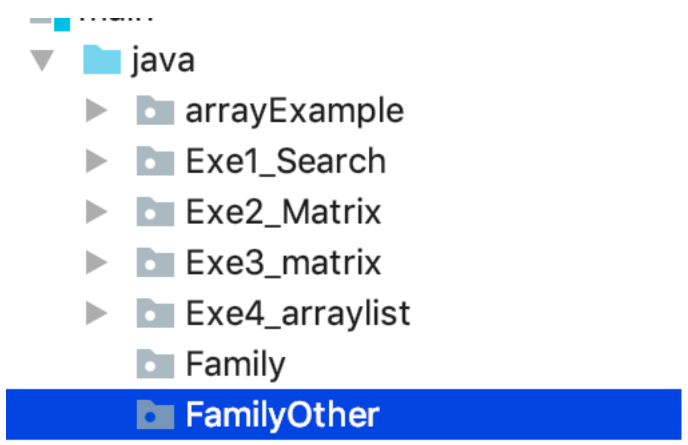
- Public: can be inherited, visible to everything
- Protected: can be inherited by sub class, visible to child class and with in package, not visible outside package
- Private: cannot be inherited, not visible to outside package and project

	Within Class	Package	Child Class	Entire Java Project
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Private	✓	✗	✗	✗

Package

- Just a folder
- To group java class that belong to one group
- A java class that has the same name cannot exist within the same package, but can exist in different package

```java
package Family;

public class Parent {

   public final String firstName;
   public final String lastName = "Alex";
   private int bankAccount = 11223344;
   private int bankAccountBalance = 1000000;

   public Parent(String firstName) {
      this.firstName = firstName;
   }

   public final void getName() {
      System.out.println(firstName + lastName);
   }

   protected void educationDirection() {
      System.out.println("Working on Medical field");
   }

   private void manageBankAccount(int input) {
      bankAccountBalance += input;
   }
}
```



java
arrayExample
Exe1_Search
Exe2_Matrix
Exe3_matrix
Exe4_arraylist
Family
FamilyOther

```java
package Family;

public class Caller {
    public static void main(String[] args) {
        Parent p = new Parent("John");
        p.educationDirection();
    }
}
```

```java
package FamilyOther;

public class Caller {
    public static void main(String[] args) {
        Parent p = new Parent("John");
        p.educationDirection(); // cannot access
    }
}
```

Child 1 that listens everything to parent

```java
package Family;

public class Child1 extends Parent{

   public Child1(String firstName) {
      super(firstName);
   }
}
```

Class refection 2

- Key word this refer to everything within current class
- Key word super reference to everything within parent class
    - super still keeps the inheritance rule
    - Cannot access private variable and method

```java
package Family;

public class Caller {
    public static void main(String[] args) {
        Child1 c1 = new Child1("Timmy");
        c1.getName();
        c1.educationDirection();
    }
}
```

```
Timmy Alex
Working on Medical field
```

Child 2 that listens nothing to parent

```java
package Family;

public class Child2 extends Parent {

    public Child2(String firstName) {
        this.firstName = firstName;
        this.lastName = "Kim";
    }

    @Override
    public void getName() {
        System.out.print(lastName + ", " + firstName);
    }

    @Override
    private void manageBankAccount(int input) {


        bankAccountBalance += input * 10;
    }

    @Override
    protected void educationDirection() {
        System.out.println("Working on Computing science");
    }
}
```

Child 2 that listens nothing to parent

```
package Family;

public class Child2 extends Parent {

    public Child2(String firstName) {
        this.firstName = firstName; // NOT allowed, have to invoke parent constructor
        this.lastName = "Kim"; // NOT allowed, family name is final
    }

    @Override
    public void getName() {
        System.out.print(lastName + ", " + firstName); // not allowed, final method is not allow to override
    }

    @Override
    private void manageBankAccount(int input) {
        // not allowed, private method is not allow to override
        // not allowed, final variable bankAccountBalance is not allow to access
        bankAccountBalance += input * 10;
    }

    @Override
    protected void educationDirection() {
        System.out.println("Working on Computing science");
    }
}
```

Child 2 corrected

```java
public class Child2 extends Parent {

    public Child2(String firstName) {
        super(firstName);
    }

    @Override
    protected void educationDirection() {
        System.out.println("Working on Computing science");
    }
}
```

```java
package Family;

public class Caller {
   public static void main(String[] args) {
      Child2 c2 = new Child2("Jimmy");
      c2.getName();
      c2.educationDirection();
   }
}
```

```
Jimmy Alex
Working on Computing science
```

```java
package Family;

public class GrandParent {

    public final String firstName;
    public final String lastName = "Alexendra";

    public GrandParent(String firstName) {
        this.firstName = firstName;
    }

    protected void singOldSongs() {
        System.out.println("Country road");
    }
}
```

```java
package Family;

public class Child1 extends Parent and GrandParent{

   public Child1(String firstName) {
      super(firstName);
   }
}
```

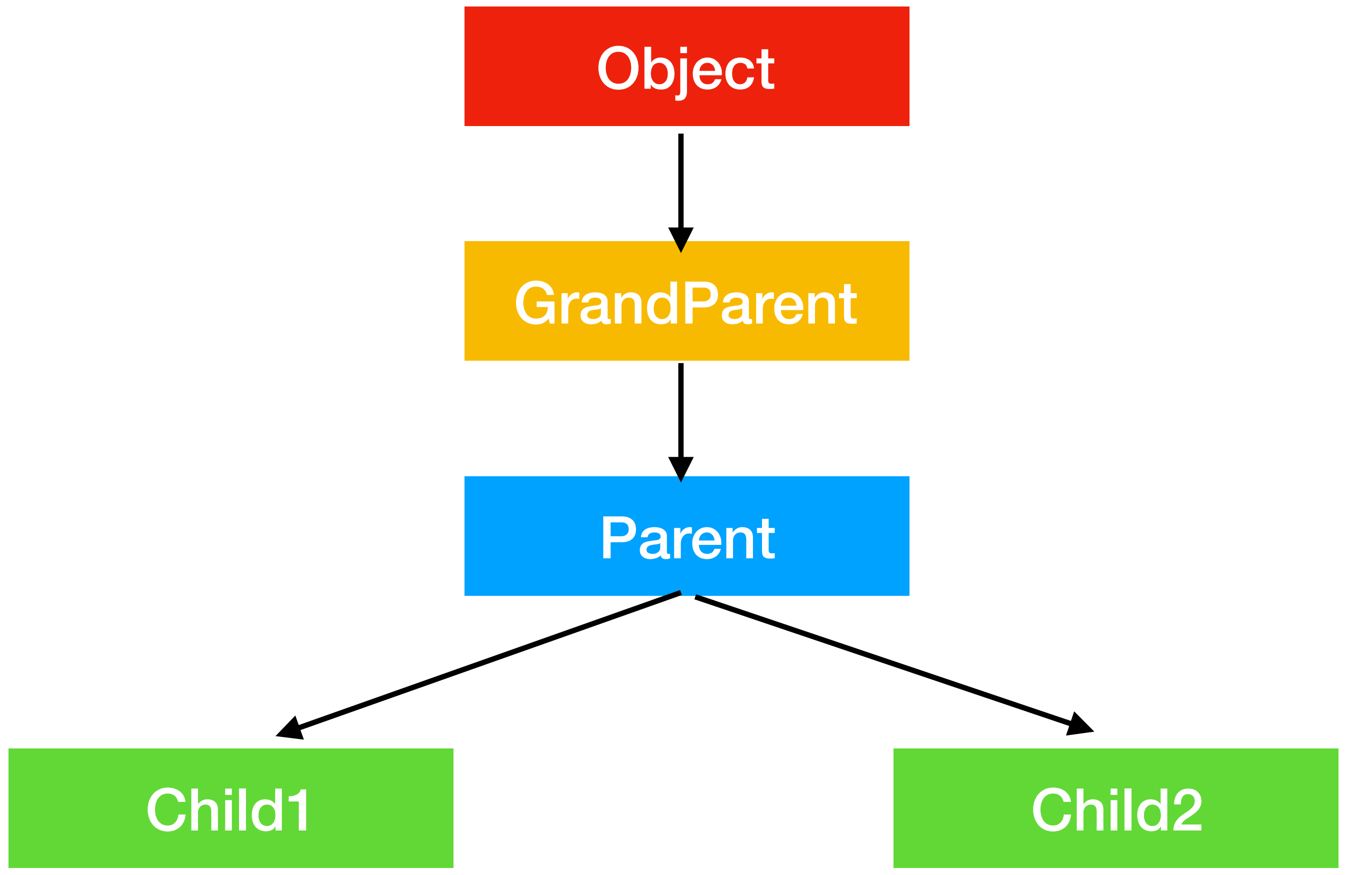
Inheritance Rule

- A parent class can be inherited by multiple child class
- But a child class can only extend only one parent class

```java
package Family;

public class Parent extends GrandParent{

   public final String lastName = "Alex";
   private int bankAccount = 11223344;
   private int bankAccountBalance = 1000000;

   public Parent(String firstName) {
      super(firstName);
   }

   public final void getName() {
      System.out.println(firstName + lastName);
   }

   protected void educationDirection() {
      System.out.println("Working on Medical field");
   }

   private void manageBankAccount(int input) {
      bankAccountBalance += input;
   }
}
```

```java
package Family;

public class Caller {
   public static void main(String[] args) {
      Child2 c2 = new Child2("Jimmy");
      c2.getName();
      c2.educationDirection();
      c2.singOldSongs();
   }
}
```

```
Jimmy Alex
Working on Computing science
Country road
```


Object
GrandParent
Parent
Child1
Child2

Child 2 with it’s own attribute, how to access this?

```java
package Family;

public class Child2 extends Parent {

    public String hobby = "Sport";

    public Child2(String firstName) {
        super(firstName);
    }

    @Override
    protected void educationDirection() {
        System.out.println("Working on Computing science");
    }
}
```

Risk of casting

Allowed

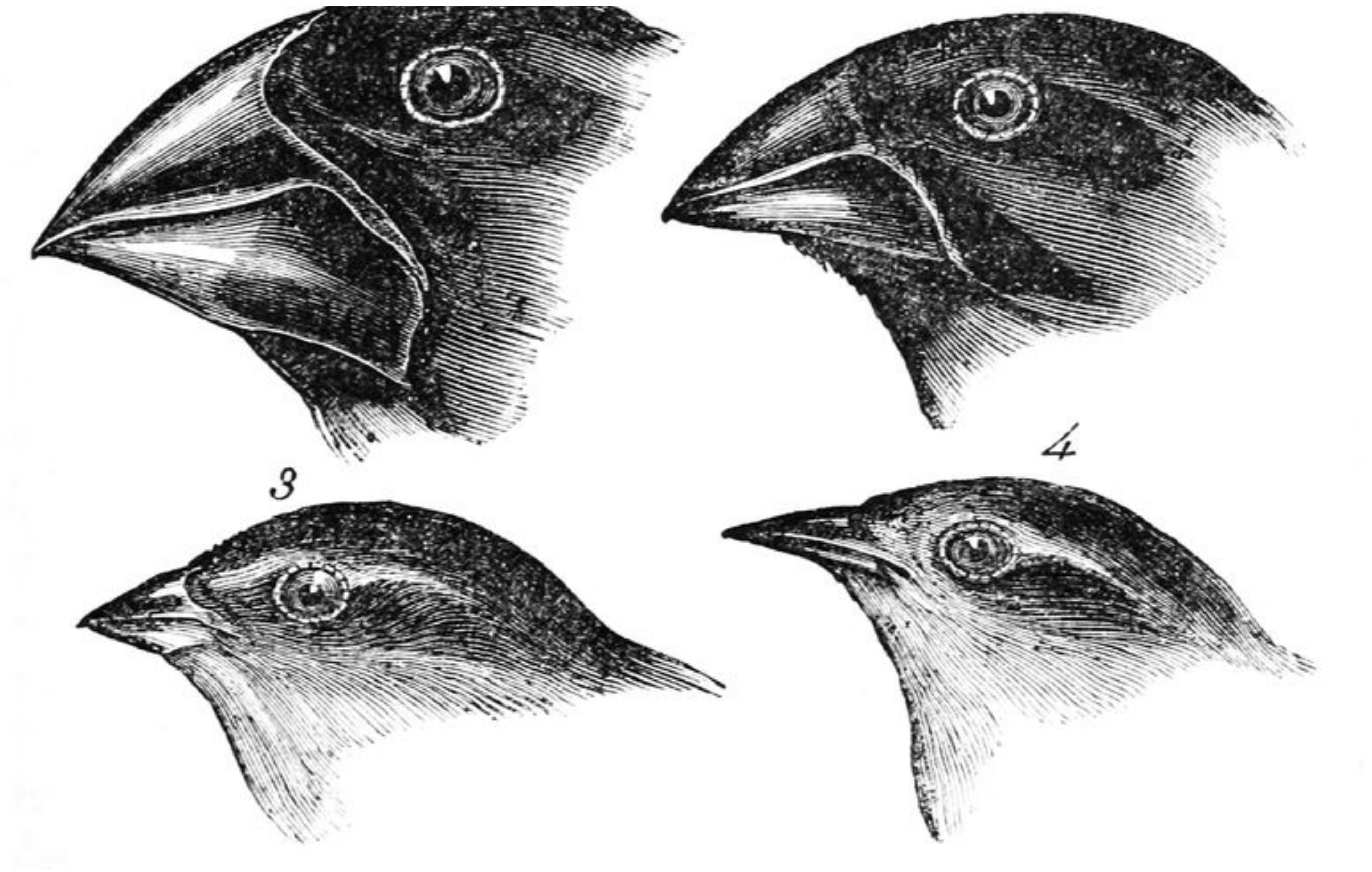
```
public class Caller {
    public static void main(String[] args) {
        Child2 c2 = new Child2("Jimmy");
        System.out.println(c2.hobby);
    }
}
```

Can’t access

```
public class Caller {
    public static void main(String[] args) {
        Parent c2 = new Child2("Jimmy");
        System.out.println(c2.hobby);
    }
}
```

Cast to the child class type to access,
However, there is a risk
Has to make sure Parent object is actually Child2

```
public class Caller {
    public static void main(String[] args) {
        Parent c2 = new Child2("Jimmy");
        System.out.println( (Child2)c2.hobby);
    }
}
```



Polymorphism

Polymorphism

- A parents class can be extended by multiple child class
- Child classes could have it’s different behaviour
- Child classes has to be describe as: one kind of parent class

```java
package produce;

public class Car {
    public String brandName;
    public final int numberWheel = 4;
    public final boolean hasBreak = true;
    public String engine = "default";
    public String branchName = "";

    public Car(String brandName, String branchName) {
        this.brandName = brandName;
        this.branchName = branchName;
    }

    @Override
    public String toString() {
        return brandName + " with " + engine + " engine";
    }

    public final String getBrandName() {
        return brandName;
    }

    public final String getBranchName() {
        return branchName;
    }

    protected void setEngine(String engine) {
        this.engine = engine;
    }
}
```

```java
package produce;

public class BMW extends Car{
    public final static String brandName = "BMW";
    public BMW (String branchName) {
        super(brandName, branchName);
    }
}
```

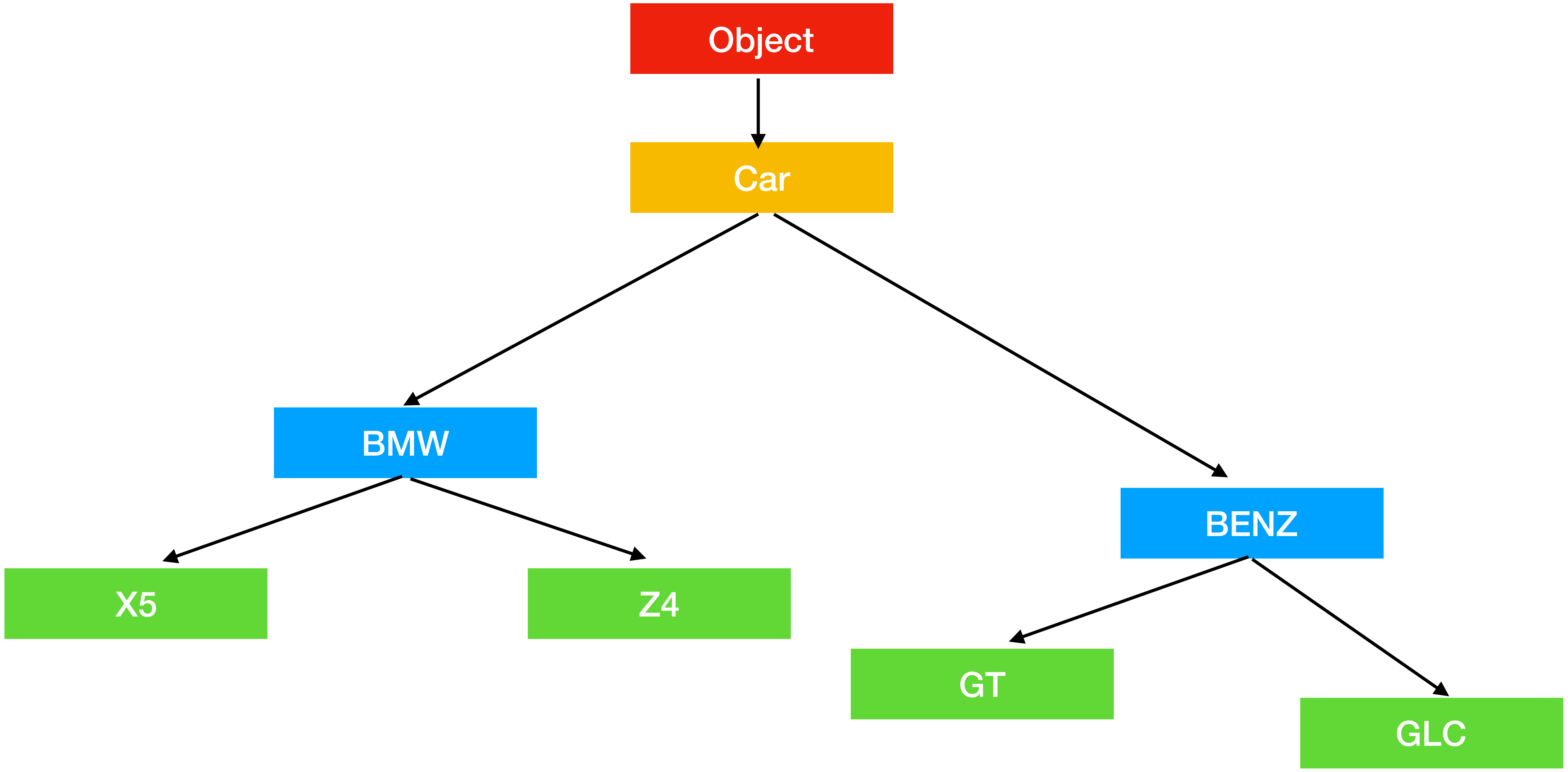
```java
package produce;

public class BENZ extends Car{
    public final static String brandName = "Benz";
    public BENZ(String branchName) {
        super(brandName, branchName);
    }
}
```

```java
package produce;

public class X5 extends BMW{
    public X5() {
        super( "X5");
        setEngine("V6");
    }

    @Override
    protected void setEngine(String engine) {
        this.engine = engine;
    }
}
```

```java
package produce;

public class GLC extends BENZ{
    public GLC() {
        super("GLC");
        setEngine("V6");
    }

    @Override
    protected void setEngine(String engine) {
        this.engine = engine;
    }
}
```

```java
package produce;

public class Z4 extends BMW{
    public Z4() {
        super( "Z5");
        setEngine("V8");
    }

    @Override
    protected void setEngine(String engine) {
        this.engine = engine;
    }
}
```

```java
package produce;

public class GT extends BENZ{
    public GT() {
        super("GT AMG");
        setEngine("V8");
    }

    @Override
    protected void setEngine(String engine) {
        this.engine = engine;
    }
}
```


Object
Car
BMW
BENZ
X5
Z4
GT
GLC

```java
package produce;

public class Factory {

    public static Car makeCar(String mode,  String branch) {
        if(mode.equals("BMW")) {
            if (branch.equals("X5")) {
                return new X5();
            } else if (branch.equals("Z4")) {
                return new Z4();
            } else {
                return null;
            }

        } else if (mode.equals("BENZ")) {
            if (branch.equals("GLC")) {
                return new GLC();
            } else if (branch.equals("GT")) {
                return new GT();
            } else {
                return null;
            }

        } else {
            return new Car("default", "default");
        }
    }
}
```

```java
package produce;

import Family.Parent;

public class Caller {
   public static void main(String[] args) {
      Car myNewCar = Factory.makeCar("BENZ", "GT");
      System.out.println(myNewCar);
   }
}
```

```
Benz GT AMG with V8 engine
```