Exercices Corrigés sur l'Héritage en Java

Exercice 1 : Héritage de Base

Objectif : Comprendre comment créer une classe dérivée en Java.

Énoncé : Créez une classe de base appelée Animal avec les attributs name et age. Créez une classe dérivée appelée Dog qui hérite de Animal et ajoute un attribut breed.

Solution :

```
class Animal {
    String name;
    int age;

    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class Dog extends Animal {
    String breed;

    Dog(String name, int age, String breed) {
        super(name, age);
```

```
        this.breed = breed;
    }

    @Override
    void display() {
        super.display();
        System.out.println("Breed: " + breed);
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog("Buddy", 3, "Golden Retriever");
        dog.display();
    }
}
```

Exercices Corrigés sur l'Héritage en Java

Exercice 2 : Utilisation de super

Objectif : Apprendre à utiliser le mot-clé super pour appeler des méthodes et des constructeurs de la classe de base.

Énoncé : Créez une méthode display dans la classe Dog qui redéfinit la méthode display de la classe Animal pour inclure l'affichage de l'attribut breed. Utilisez super pour appeler la méthode display de la classe de base.

Solution :

```
class Animal {
    String name;
    int age;

    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class Dog extends Animal {
    String breed;
```

```
    Dog(String name, int age, String breed) {
        super(name, age);
        this.breed = breed;
    }

    @Override
    void display() {
        super.display();
        System.out.println("Breed: " + breed);
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog("Buddy", 3, "Golden Retriever");
        dog.display();
    }
}
```

Exercices Corrigés sur l'Héritage en Java

Exercice 3 : Polymorphisme

Objectif : Comprendre le polymorphisme à travers l'héritage.

Énoncé : Créez une classe Cat qui hérite de Animal et ajoute un attribut color. Créez un tableau d'Animal qui contient des objets Dog et Cat, puis affichez les informations de chaque objet en utilisant une boucle.

Solution :

```
class Animal {
    String name;
    int age;

    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class Dog extends Animal {
    String breed;

    Dog(String name, int age, String breed) {
```

```
        super(name, age);
        this.breed = breed;
    }

    @Override
    void display() {
        super.display();
        System.out.println("Breed: " + breed);
    }
}

class Cat extends Animal {
    String color;

    Cat(String name, int age, String color) {
        super(name, age);
        this.color = color;
    }

    @Override
    void display() {
        super.display();
        System.out.println("Color: " + color);
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Animal[] animals = {
            new Dog("Buddy", 3, "Golden Retriever"),
            new Cat("Whiskers", 2, "Black")
        };

        for (Animal animal : animals) {
            animal.display();
            System.out.println();
        }
    }
}
```

Exercices Corrigés sur l'Héritage en Java

Exercice 4 : Héritage Multiple par Interfaces

Objectif : Comprendre comment réaliser un héritage multiple en utilisant des interfaces.

Énoncé : Créez une interface Pet avec une méthode play(). Créez une classe Fish qui hérite de Animal et implémente Pet. Implémentez la méthode play() pour afficher un message indiquant que le poisson joue.

Solution :

```
interface Pet {
    void play();
}

class Animal {
    String name;
    int age;

    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

```
class Fish extends Animal implements Pet {
    Fish(String name, int age) {
        super(name, age);
    }

    @Override
    public void play() {
        System.out.println(name + " is playing in the water.");
    }

    @Override
    void display() {
        super.display();
        play();
    }
}

public class Main {
    public static void main(String[] args) {
        Fish fish = new Fish("Nemo", 1);
        fish.display();
    }
}
```