우리와 새장

동물원에서 범용으로 쓸 수 있는 우리와 새장의 템플릿을 만들고자 합니다. 주어진 코드에 Generic을 적용해서 사용할 수 있도록 하시오.

- 문제

주어진 코드의 TODO를 해결하시오.

(1) : Cage 클래스와 Container 클래스에 Generic을 적용하시오. 이 때, 클래스 내부의 멤버 함수는 수정하지 않아야 합니다.

(2) : (1)에서 작성한 클래스를 사용해서, landContainer 변수가 'output' 블록의 작업을 수행할 수 있도록 수정하시오. 이 때 'output' 블록은 수정하지 않아야 합니다.

(3) : 'Penguin' 블록은 컴파일 오류가 발생하고 그 위의 'Duck' 블록은 작동하도록 birdContainer 변수의 타입을 수정하시오. 수정한 후, Domjudge 업로드를 위해 오류가 발생하는 코드를 주석처리하시오.

- 힌트

'TYPE extends (Class)', 'TYPE super (Class)'를 잘 이용해야 합니다.

- 주어진 코드

```
// Example of class inheritance
class Animal {
	private final String name;
	private final int age;

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

	public String info() {
		return this.getClass().getSimpleName() + " " + name + " (" + age + ")";
	}
}


class LandAnimal extends Animal {
	public LandAnimal(String name, int age) {
		super(name, age);
	}

	public String bark() {return "";}
}


class Dog extends LandAnimal {
	public Dog(String name, int age) {
		super(name, age);
	}

	public String bark() {return "bow!";}
}


class Bear extends LandAnimal {
```

```
        public Bear(String name, int age) {
                super(name, age);
        }

        public String bark() {return "roar!";}
}


class Bird extends Animal {
        public Bird(String name, int age) {
                super(name, age);
        }

        public boolean canFly() {return true;}
}


class Duck extends Bird {
        public Duck(String name, int age) {
                super(name, age);
        }
}


class Penguin extends Bird {
        public Penguin(String name, int age) {
                super(name, age);
        }

        public boolean canFly() {return false;}
}
//End of Example


//TODO (1) : Make Container class working
//Hint : Use upper bound
class Cage<T> {
        private T animal;


        public String info() {
                return "Cage : " + animal.info();
        }


        public void putAnimal(T animal) {
                this.animal = animal;
        }


        public T getAnimal() {
                return animal;
        }
}


// TODO (1) : Make Container class working
// Hint : Use upper bound
class Container<T> {
        private T cage;


        public String info() {
                return "Container : " + cage.info();
        }
```

```
	public void putCage(T cage) {
		this.cage = cage;
	}


	public T getCage() {
		return cage;
	}
}


public class Zoo {
	public static void main(String args[]) {

		Dog dog = new Dog("Pluto", 10);
		Bear bear = new Bear("Bongo", 15);
		Duck duck = new Duck("Donald", 4);
		Penguin penguin = new Penguin("Pablo", 6);


		// TODO (2) : make block 'Output' working
		// Hint : modify type of landContainer variable
		Container<Cage<LandAnimal>> landContainer = new Container<>();


		// Block 'output'
		// Do not modify this block
		Cage<Bear> bearCage = new Cage<>();
		Cage<Dog> dogCage = new Cage<>();


		bearCage.putAnimal(bear);
		landContainer.putCage(bearCage);


		System.out.println(landContainer.info());
		System.out.println(landContainer.getCage().getAnimal().bark());


		dogCage.putAnimal(dog);
		landContainer.putCage(dogCage);


		System.out.println(landContainer.info());
		System.out.println(landContainer.getCage().getAnimal().bark());
		// Block 'output' end


		// TODO (3) : make Block 'Penguin' cannot be compiled
		// Hint : modify type of birdContainer using lower bound
		Container<Cage<?>> birdContainer = new Container<>();


		Cage<Penguin> penguinCage = new Cage<Penguin>();
		Cage<Bird> birdCage = new Cage<Bird>();


		// TODO (3) : Block 'Duck'
		birdCage.putAnimal(duck);
		birdContainer.putCage(birdCage);
		// Block end
```

```
                System.out.println(birdContainer.getCage().getAnimal().info());


                // TODO (3) : Block 'Penguin' : Make this block cannot be compiled
                // Make this block to comment to submit on DomJudge
                penguinCage.putAnimal(penguin);
                birdContainer.putCage(penguinCage);
                // Block end


                System.out.println(birdContainer.getCage().getAnimal().info());

        }
}
```

- 출력

```
Container : Cage : Bear Bongo (15)
roar!
Container : Cage : Dog Pluto (10)
bow!
Duck Donald (4)
Duck Donald (4)
```