```
// AttackStrategy.java
public interface AttackStrategy {
    void attack();
}

// DefenseStrategy.java
public interface DefenseStrategy {
    void defend();
}
// CastSpell.java
public class CastSpell implements AttackStrategy {
    public void attack() {
        System.out.println("Wizard casts a spell!");
    }
}

// ShootArrow.java
public class ShootArrow implements AttackStrategy {
    public void attack() {
        System.out.println("Archer shoots an arrow!");
    }
}

// SwingSword.java
public class SwingSword implements AttackStrategy {
    public void attack() {
        System.out.println("Knight attacks with a sword!");
    }
}
// Shield.java
public class Shield implements DefenseStrategy {
    public void defend() {
        System.out.println("Using a shield to defend!");
    }
}

// Dodge.java
public class Dodge implements DefenseStrategy {
    public void defend() {
        System.out.println("Dodging to avoid the attack!");
    }
}
// CreateMagic.java
public class CreateMagic implements DefenseStrategy {
    public void defend() {
        System.out.println("Creating a magic barrier for defense!");
    }
}
// Character.java
public abstract class Character {
    private AttackStrategy attackStrategy;
    private DefenseStrategy defenseStrategy;
```

```
public Character(AttackStrategy attackStrategy, DefenseStrategy defenseStrategy) {
        this.attackStrategy = attackStrategy;
        this.defenseStrategy = defenseStrategy;
    }

    public void setAttackStrategy(AttackStrategy attackStrategy) {
        this.attackStrategy = attackStrategy;
    }

    public void setDefenseStrategy(DefenseStrategy defenseStrategy) {
        this.defenseStrategy = defenseStrategy;
    }

    public void performAttack() {
        attackStrategy.attack();
    }

    public void performDefense() {
        defenseStrategy.defend();
    }
}
// Knight.java
public class Knight extends Character {
    public Knight() {
        super(new SwingSword(), new Shield());
    }
}
// Wizard.java
public class Wizard extends Character {
    public Wizard() {
        super(new CastSpell(), new CreateMagic());
    }
}
// Archer.java
public class Archer extends Character {
    public Archer() {
        super(new ShootArrow(), new Dodge());
    }
}
public class GameApp {
    public static void main(String[] args) {
        Character knight = new Knight();
        knight.performAttack(); // Knight attacks with a sword!
        knight.performDefense(); // Using a shield to defend!

        Character wizard = new Wizard();
        wizard.performAttack(); // Wizard casts a spell!
        wizard.performDefense(); // Creating a magic barrier for defense!

        Character archer = new Archer();
        archer.performAttack(); // Archer shoots an arrow!
        archer.performDefense(); // Dodging to avoid the attack!
    }
}
```

Problem Scenario:

In the GameApp, we have three types of characters:

1. Knight: Attacks with a sword and has three different strategies for defense:
    - Shield
    - Dodge
    - Magic Barrier
2. Wizard: Attacks by casting spells and defends using the magic barrier.
3. Archer: Attacks by shooting arrows and defends by dodging.

The goal is to implement two types of Strategy:

- DefenseStrategy, with the following specific strategies:
    - Shield: Protects with a shield.
    - Dodge: Evades the attack.
    - CreateMagic: Creates a magical barrier to defend.
- AttackStrategy, with the following specific strategies:
    - CastSpell: Casts a spell as an attack.
    - ShootArrow: Shoots arrows as an attack.
    - SwingSword: Swings a sword as an attack.