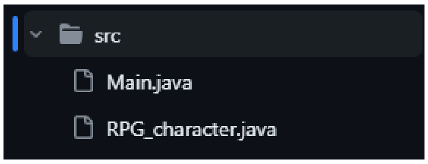
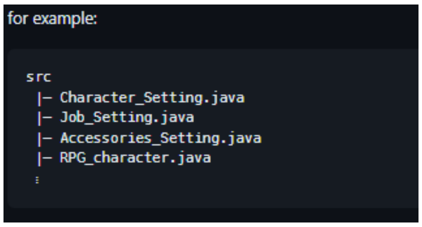
What can be improved?

- improve by separating every public class into each file.



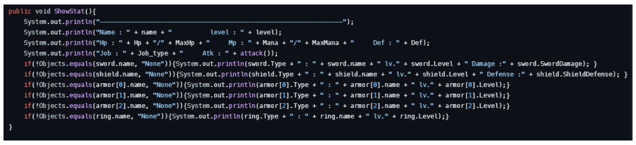
```
src
    Main.java
    RPG_character.java
```



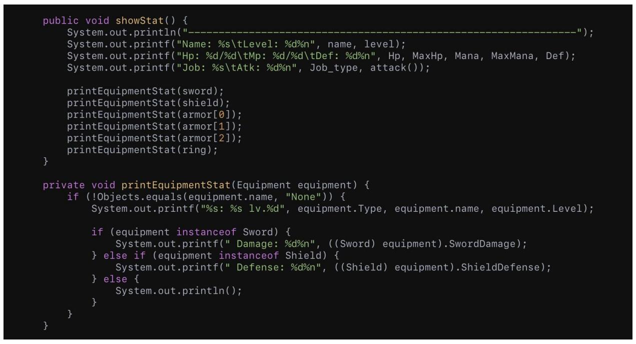
for example:

```
src
 |- Character_Setting.java
 |- Job_Setting.java
 |- Accessories_Setting.java
 |- RPG_character.java
 ⋮
```

- improve showstat() method in RPG_character.java



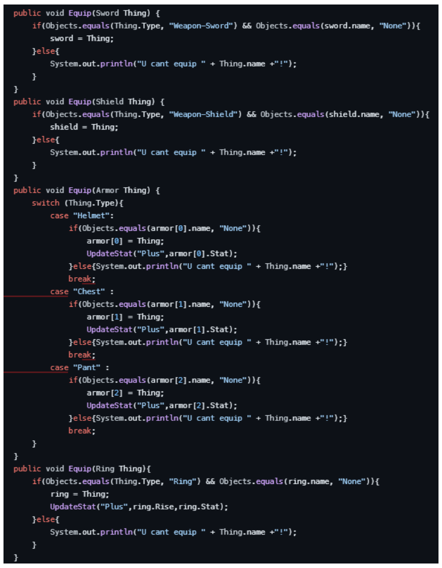
```
public void ShowStat(){
    System.out.println("----------------------------------------------------------------");
    System.out.println("Name : " + name + "          level : " + level);
    System.out.println("Hp : " + Hp + "/" + MaxHp + "     Mp : " + Mana + "/" + MaxMana + "     Def : " + Def);
    System.out.println("Job : " + Job_type + "     Atk : " + attack());
    if(!Objects.equals(sword.name, "None")){System.out.println(sword.Type + " : " + sword.name + " lv." + sword.Level + " Damage :" + sword.SwordDamage); }
    if(!Objects.equals(shield.name, "None")){System.out.println(shield.Type + " : " + shield.name + " lv." + shield.Level + " Defense :" + shield.ShieldDefense); }
    if(!Objects.equals(armor[0].name, "None")){System.out.println(armor[0].Type + " : " + armor[0].name + " lv." + armor[0].Level);}
    if(!Objects.equals(armor[1].name, "None")){System.out.println(armor[1].Type + " : " + armor[1].name + " lv." + armor[1].Level);}
    if(!Objects.equals(armor[2].name, "None")){System.out.println(armor[2].Type + " : " + armor[2].name + " lv." + armor[2].Level);}
    if(!Objects.equals(ring.name, "None")){System.out.println(ring.Type + " : " + ring.name + " lv." + ring.Level);}
}
```



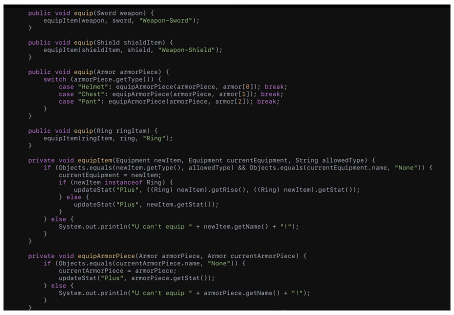
```
    public void showStat() {
        System.out.println("------------------------------------------------------------------");
        System.out.printf("Name: %s\tLevel: %d%n", name, level);
        System.out.printf("Hp: %d/%d\tMp: %d/%d\tDef: %d%n", Hp, MaxHp, Mana, MaxMana, Def);
        System.out.printf("Job: %s\tAtk: %d%n", Job_type, attack());

        printEquipmentStat(sword);
        printEquipmentStat(shield);
        printEquipmentStat(armor[0]);
        printEquipmentStat(armor[1]);
        printEquipmentStat(armor[2]);
        printEquipmentStat(ring);
    }

    private void printEquipmentStat(Equipment equipment) {
        if (!Objects.equals(equipment.name, "None")) {
            System.out.printf("%s: %s lv.%d", equipment.Type, equipment.name, equipment.Level);

            if (equipment instanceof Sword) {
                System.out.printf(" Damage: %d%n", ((Sword) equipment).SwordDamage);
            } else if (equipment instanceof Shield) {
                System.out.printf(" Defense: %d%n", ((Shield) equipment).ShieldDefense);
            } else {
                System.out.println();
            }
        }
    }
```

This code can be improved for cleaner formatting, introduces a separate method for printing equipment stats, and utilizes polymorphism to differentiate between sword and shield equipment.

- improve Equip and method



```java
public void Equip(Sword Thing) {
    if(Objects.equals(Thing.Type, "Weapon-Sword") && Objects.equals(sword.name, "None")){
        sword = Thing;
    }else{
        System.out.println("U cant equip " + Thing.name +"!");
    }
}
public void Equip(Shield Thing) {
    if(Objects.equals(Thing.Type, "Weapon-Shield") && Objects.equals(shield.name, "None")){
        shield = Thing;
    }else{
        System.out.println("U cant equip " + Thing.name +"!");
    }
}
public void Equip(Armor Thing) {
    switch (Thing.Type){
        case "Helmet":
            if(Objects.equals(armor[0].name, "None")){
                armor[0] = Thing;
                UpdateStat("Plus",armor[0].Stat);
            }else{System.out.println("U cant equip " + Thing.name +"!");}
            break;
        case "Chest" :
            if(Objects.equals(armor[1].name, "None")){
                armor[1] = Thing;
                UpdateStat("Plus",armor[1].Stat);
            }else{System.out.println("U cant equip " + Thing.name +"!");}
            break;
        case "Pant" :
            if(Objects.equals(armor[2].name, "None")){
                armor[2] = Thing;
                UpdateStat("Plus",armor[2].Stat);
            }else{System.out.println("U cant equip " + Thing.name +"!");}
            break;
    }
}
public void Equip(Ring Thing){
    if(Objects.equals(Thing.Type, "Ring") && Objects.equals(ring.name, "None")){
        ring = Thing;
        UpdateStat("Plus",ring.Rise,ring.Stat);
    }else{
        System.out.println("U cant equip " + Thing.name +"!");
    }
}
```



```java
    public void equip(Sword weapon) {
        equipItem(weapon, sword, "Weapon-Sword");
    }

    public void equip(Shield shieldItem) {
        equipItem(shieldItem, shield, "Weapon-Shield");
    }

    public void equip(Armor armorPiece) {
        switch (armorPiece.getType()) {
            case "Helmet": equipArmorPiece(armorPiece, armor[0]); break;
            case "Chest": equipArmorPiece(armorPiece, armor[1]); break;
            case "Pant": equipArmorPiece(armorPiece, armor[2]); break;
        }
    }

    public void equip(Ring ringItem) {
        equipItem(ringItem, ring, "Ring");
    }

    private void equipItem(Equipment newItem, Equipment currentEquipment, String allowedType) {
        if (Objects.equals(newItem.getType(), allowedType) && Objects.equals(currentEquipment.name, "None")) {
            currentEquipment = newItem;
            if (newItem instanceof Ring) {
                updateStat("Plus", ((Ring) newItem).getRise(), ((Ring) newItem).getStat());
            } else {
                updateStat("Plus", newItem.getStat());
            }
        } else {
            System.out.println("U can't equip " + newItem.getName() + "!");
        }
    }

    private void equipArmorPiece(Armor armorPiece, Armor currentArmorPiece) {
        if (Objects.equals(currentArmorPiece.name, "None")) {
            currentArmorPiece = armorPiece;
            updateStat("Plus", armorPiece.getStat());
        } else {
            System.out.println("U can't equip " + armorPiece.getName() + "!");
        }
    }
```

can improve by reducing code duplication. It also leverages polymorphism to handle specific cases for rings. Additionally, it improves method and parameter naming for better readability.