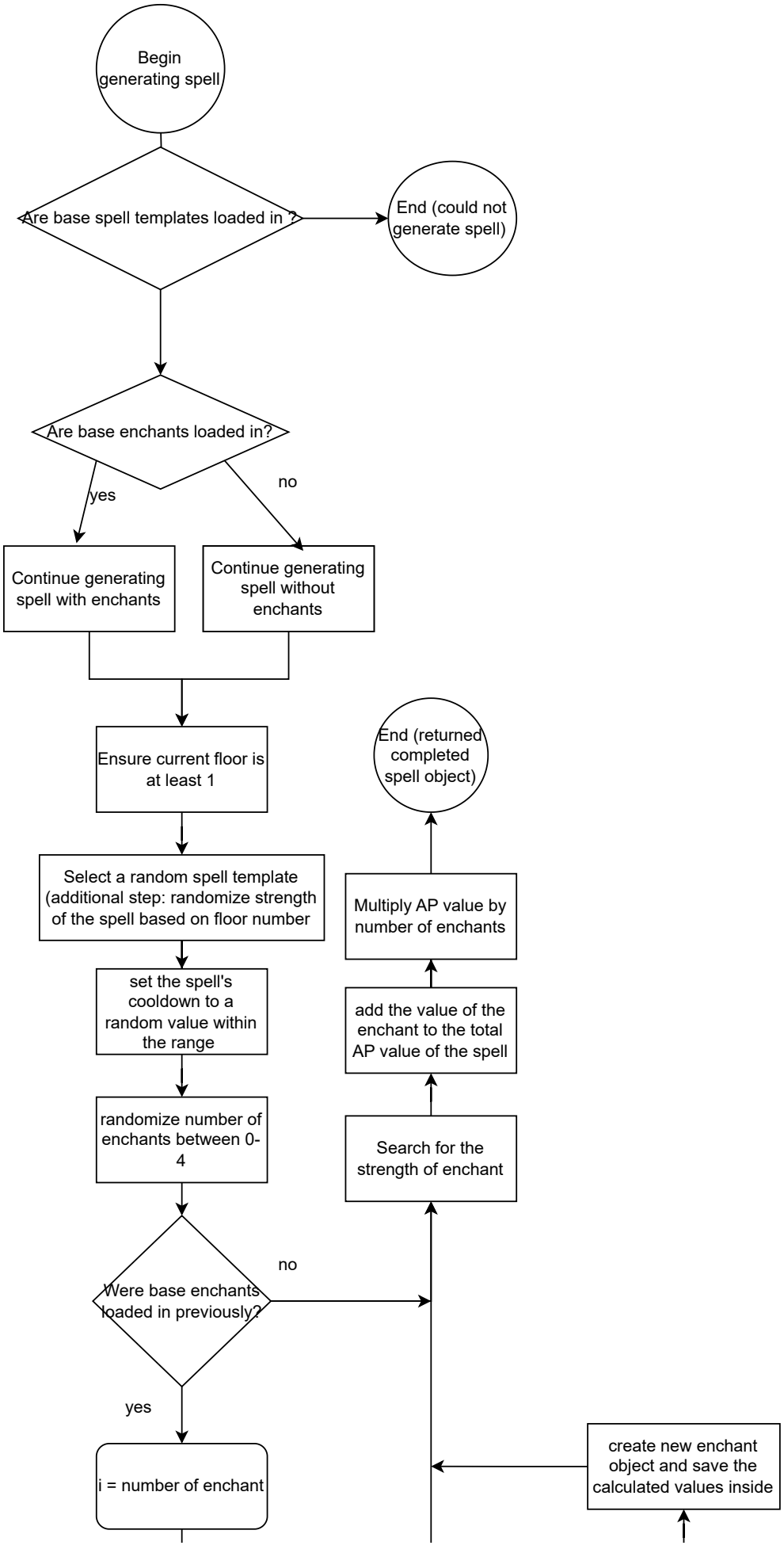
```
/**
 * Generates a random player spell bas
 * @param currentFloor The current flo
 * @return A fully generated PlayerSpe
 */
public PlayerSpell generateSpellDrop(i
    if (allBaseSpellTemplates == null
        System.err.println("SpellGener
        return null;
    }
    if (allPossibleBaseEnchants == nul
        System.err.println("SpellGener
    }
    if (currentFloor < 1) {
        currentFloor = 1; // Ensure fl
    }

    // 1. Select Random Base Spell Ter
    BaseSpellTemplate selectedTemplate

    // 2. Randomize Cooldown
```

```
sed on the current floor.
oor number (1-10).
ell object, or null if generation fails.

int currentFloor) {
 || allBaseSpellTemplates.isEmpty()) {
rator Error: No base spell templates loaded!");


ll) {
rator Warning: No base enchants loaded. Spells will have no enchants.");


loor is at least 1


mplate
e = allBaseSpellTemplates.get(random.nextInt(allBaseSpellTemplates.size()));
```



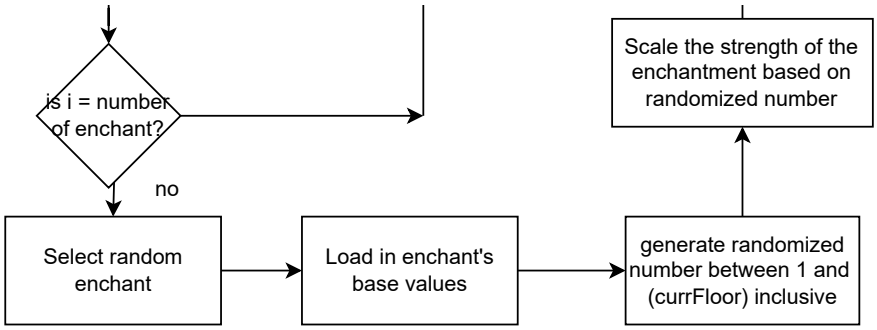

```
// 2. Randomize Cooldown
double minCD = selectedTemplate.co
double maxCD = selectedTemplate.co
double rawCooldown = minCD + (maxC
double actualCooldownSeconds = Big

// 3. Randomize Spell Core Effect
int M_spell = random.nextInt(curre
double effectiveCoreEffectValue =

// 4. Determine Number of Enchant
int numberOfEnchants = random.nex
List<EnchantInstance> appliedEnch

// 5. Generate Enchant Instances
if (allPossibleBaseEnchants != nu
    for (int i = 0; i < numberOfE
        BaseEnchant selectedBaseE

        double trueBaseValue = se
        double maxTotalValueAtFlo
        double maxPotentialIncrea

        int M_enchant = random.ne

        // Strength is calculated
        double finalEnchantValue


        finalEnchantValue = Math.
        finalEnchantValue = Math.

        // Create EnchantInstance
        appliedEnchants.add(new E
    }
}

// 6. Calculate Final AP Cost (ba
double apSum = GLOBAL_BASE_AP_COS

CoreEffectData coreEffect = sele
// Check if the core effect type
// that should contribute to AP
switch (coreEffect.type()) {
    case DAMAGE:
    case HEALING:
    case SHIELD_APPLICATION:
    case APPLY_DOT:
    case BUFF_PLAYER:
    case DEBUFF_ENEMY:
        apSum += Math.abs(effect
        break;
    default:
        // For other core effect
        // or shouldn't contribu
        System.out.println("Spel
                           " doe
        break;
}

// Add enchant values to AP sum
for (EnchantInstance enchant : a
    // For percentage enchants,
    // For flat value enchants,
    apSum += Math.abs(enchant.fi
}

double enchantMultiplier = ENCHA
int finalAPCost = (int) Math.rou
finalAPCost = Math.max(5, finalA

// 7. Construct and
return new PlayerSp
        selectedTem
        actualCoold
        effectiveCo
```

```
ooldownRange().minSeconds();
ooldownRange().maxSeconds();
CD - minCD) * random.nextDouble();
gDecimal.valueOf(rawCooldown).setScale(1, RoundingMode.HALF_UP).doubleValue();

 Strength
entFloor) + 1; // M is 1 to currentFloor
 selectedTemplate.coreEffect().baseValue() * M_spell;
```

```
ts (0-4)
xtInt(5);
hants = new ArrayList<>();


ull && !allPossibleBaseEnchants.isEmpty() && numberOfEnchants > 0) {
Enchants; i++) {
Enchant = allPossibleBaseEnchants.get(random.nextInt(allPossibleBaseEnchants.size()));

electedBaseEnchant.trueBaseValue();
oor50 = selectedBaseEnchant.maxTotalValueAtFloor50();
aseByF50 = maxTotalValueAtFloor50 - trueBaseValue;

extInt(currentFloor) + 1; // M is 1 to currentFloor

d based on M_enchant (effective floor)
 = trueBaseValue +
   Math.round(((maxPotentialIncreaseByF50 / 50.0) * M_enchant));

.min(finalEnchantValue, maxTotalValueAtFloor50);
.max(finalEnchantValue, trueBaseValue);

e without mValueUsed, as it's not needed for AP cost now
EnchantInstance(selectedBaseEnchant, finalEnchantValue));



ased on effectiveCoreEffectValue and finalRolledValue of enchants)
ST;
```

```
ectedTemplate.coreEffect();
e is one that has a directly quantifiable 'strength'
 cost.









tiveCoreEffectValue);


t types that might not have a simple numerical "strength"
ute to AP cost from their core effect, do nothing here.
llGenerator Note: CoreEffectType " + coreEffect.type() +
es not currently have a defined AP cost contribution for its core effect.");




appliedEnchants) {
 add the number (e.g., 25 for 25%)
 add the flat value.
inalRolledValue());


ANT_COUNT_AP_MULTIPLIERS[Math.min(numberOfEnchants, ENCHANT_COUNT_AP_MULTIPLIERS.ler
und(apSum * enchantMultiplier);
APCost); // Ensure AP cost is at least 5
```

```
 Return PlayerSpell Instance
ell(
plate,
ownSeconds,
reEffectValue,
```




```
            appliedEnch
            finalAPCost
);
```

```
ants,
```