1. COUNTER

```
class Counter:
    def __init__(self):
        self.value = 0

    def reset(self):
        self.value = 0

    def click(self):
        self.value += 1

    def getValue(self):
        return self.value

# Penggunaan
tally = Counter()
tally.reset()
tally.click()
tally.click()
result = tally.getValue()  # Result is 2
print(result)
tally.click()
result = tally.getValue()  # Result is 3
print(result)
```

Mengapa jika perintah `tally.reset()` (line 2 di atas) tidak dijalankan (dihapus), menyebabkan error?

= Jika `tally.reset()` dihapus, sebenarnya tidak akan menyebabkan error. Nilai awal `self.value` sudah diset ke 0 di `__init__`. Jadi menghapus `tally.reset()` tidak akan menyebabkan error, tapi mungkin bisa menyebabkan perilaku yang tidak diinginkan jika `Counter` sudah digunakan sebelumnya.

2. REGISTER

```
class CashRegister:
    def __init__(self):
        self.total = 0
        self.count = 0

    def addItem(self, price):
        self.total += price
        self.count += 1

    def getTotal(self):
        return round(self.total, 2)

    def getCount(self):
        return self.count

# Penggunaan
```

```
register1 = CashRegister()
register1.addItem(1.95)
register1.addItem(0.95)
register1.addItem(2.50)

register2 = CashRegister()
register2.addItem(3.75)
register2.addItem(0.15)
register2.addItem(2.25)
register2.addItem(1.80)

# Mencetak total
print(f"Register 1 sells {register1.getCount()} items, with the total
amount of ${register1.getTotal()}")
print(f"Register 2 sells {register2.getCount()} items, with the total
amount of ${register2.getTotal()}")
```

3. RATIONAL NUMBER

```
class Fraction:
    def __init__(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero")
        self.numerator = numerator
        self.denominator = denominator
        self._simplify()

    def _simplify(self):
        # Fungsi bantuan untuk menyederhanakan pecahan
        def gcd(a, b):
            while b:
                a, b = b, a % b
            return a

        common = gcd(abs(self.numerator), abs(self.denominator))
        self.numerator //= common
        self.denominator //= common

        if self.denominator < 0:
            self.numerator = -self.numerator
            self.denominator = -self.denominator

    def getRationalForm(self):
        return f"{self.numerator}/{self.denominator}"

    def getFloatingPointValue(self):
        return self.numerator / self.denominator

# Penggunaan
frac = Fraction(1, 3)
print(f"Rational form: {frac.getRationalForm()}")
print(f"Floating point value: {frac.getFloatingPointValue()}")
```