Refactoring 3

- Estudiante: Javier López
- Problema: Problema 1

Código Inicial

```
const plays = require("./data/plays.json");
const invoices = require("./data/invoices.json");

function statement (invoice, plays) {
  let totalAmount = 0;
  let volumeCredits = 0;
  let result = `Statement for ${invoice.customer}\n`;
  const format = new Intl.NumberFormat("en-US",
                        { style: "currency", currency: "USD'
                          minimumFractionDigits: 2 }).forma
  for (let perf of invoice.performances) {
    const play = plays[perf.playID];
    let thisAmount = 0;

    switch (play.type) {
    case "tragedy":
      thisAmount = 40000;
      if (perf.audience > 30) {
        thisAmount += 1000 * (perf.audience - 30);
      }
      break;
    case "comedy":
      thisAmount = 30000;
      if (perf.audience > 20) {
        thisAmount += 10000 + 500 * (perf.audience - 20);
      }
      thisAmount += 300 * perf.audience;
      break;
    default:
        throw new Error(`unknown type: ${play.type}`);
    }

    // add volume credits
    volumeCredits += Math.max(perf.audience - 30, 0);
    // add extra credit for every ten comedy attendees
    if ("comedy" === play.type) volumeCredits += Math.floor(

    // print line for this order
    result += `  ${play.name}: ${format(thisAmount/100)} ($
    totalAmount += thisAmount;
  }
  result += `Amount owed is ${format(totalAmount/100)}\n`;
  result += `You earned ${volumeCredits} credits\n`;
  return result;
}

console.log(statement(invoices[0], plays));
```

Solución

```
const plays = require("./data/plays.json");
const invoices = require("./data/invoices.json");

const formatAmount = new Intl.NumberFormat("en-US", {
  style: "currency",
  currency: "USD",
  minimumFractionDigits: 2,
}).format;

function performanceAmounts(performance) {
  let amount = 0;
  let volumeCredits = Math.max(performance.audience - 30, 0
  switch (performance.play.type) {
    case "tragedy":
      amount = 40000;
      if (performance.audience > 30) {
        amount += 1000 * (performance.audience - 30);
      }
      break;
    case "comedy":
      amount = 30000;
      if (performance.audience > 20) {
        amount += 10000 + 500 * (performance.audience - 20)
      }
      amount += 300 * performance.audience;
      volumeCredits += Math.floor(performance.audience / 5)
      break;
    default:
      throw new Error(`unknown type: ${performance.play.type
  }
  return { amount, volumeCredits };
}

function buildStatementData(invoice, plays) {
  const result = {
    customer: invoice.customer,
    performances: invoice.performances.map((perf) => {
      const play = plays[perf.playID];
      return {
        ...perf,
        play,
      };
    }),
    lineItems: [],
    values: {
      totalAmount: 0,
      totalVolumeCredits: 0,
    },
  };

  for (let perf of result.performances) {
    const { amount, volumeCredits } = performanceAmounts(pe
    const item = {
      performance: perf,
      amount: amount,
    };
    result.values.totalAmount += amount;
    result.values.totalVolumeCredits += volumeCredits;
    result.lineItems.push(item);
  }

  return result;
}

function renderStatementData(data) {
  let result = [];
  result.push(`Statement for ${data.customer}`);
  for (let { performance, amount } of data.lineItems) {
    result.push(
      `  ${performance.play.name}: ${formatAmount(amount / 1
      } seats)`,
    );
  }
  result.push(`Amount owed is ${formatAmount(data.values.tot
  result.push(`You earned ${data.values.totalVolumeCredits}

  return result.join("\n");
}

function statement(invoice, plays) {
  return renderStatementData(buildStatementData(invoice, pla
}
console.log(statement(invoices[0], plays));
```

Refactorings Utilizados

Extract Function

Buscando la cohesión se utiliza la extracción de varias funciones:

- formatAmount: Se extrae la función formatAmount para formatear los montos
- performanceAmounts: Se extrae la lógica para determinar el monto y los créditos de una performance.

- buildStatementData: Se enriquece los datos iniciales para ser usados por las demás funciones transformadoras.
- renderStatementData: Se encarga de renderizar los datos, pasarlos de una estructura a un string en este caso.

Esto mejora la mantenibilidad al tener funciones más pequeñas y con responsabilidades más acotadas.

Introduce Parameter Object

- La función buildStatementData recibe dos parámetros, invoice y plays, se introduce un objeto que contenga ambos parámetros para mejorar la legibilidad del código. Este objeto además se enriquece con los lineItems y los valores de totalAmount y totalVolumeCredits.

Este refactor permite que las diferentes funciones tengan menos parametros y los datos en un formato fácil para realizar los procedimientos, lo que mejora la mantenibilidad del código.