Shopping Cart Implementation Exercise

Overview

Improve the provided code snippet, which outlines a basic e-commerce shopping cart operation. The code handles adding items to a cart and calculating the total price. Your task is to suggest improvements that enhance the code's readability, maintainability, efficiency, and correctness whilst preserving the existing business functionality.

Feel free to use a programming language you feel comfortable with.

Out of scope:

- Security is not part of this exercise
- The code snippet does not need to compile and run
- You can omit unit/integration testing for this piece of code
- Database implementation details

Please submit:

- An outline of your approach, ensuring that the fundamental business capability remains unchanged
- Brief explanation of your implementation choices
- The improved code snippet
- Document any unclear library usage

Estimated time:

1 to 1.5 hours

Remember: We're not expecting architectural redesigns or advanced patterns. Focus on making the code cleaner, more maintainable, and correct.

Original Code:

```java
import org.springframework.web.bind.annotation.*;
import java.math.BigDecimal;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/shop")
public class ShoppingCartController {

    // Store carts in memory (in real app, this would be a database)
    private Map<String, Map<String, Object>> carts = new HashMap<>();

    @PostMapping("/addItem")
    public String addItem(@RequestParam("cartId") String cartId,
                          @RequestParam("itemName") String itemName,
                          @RequestParam("price") double price,
                          @RequestParam("quantity") int quantity) {

        // Get or create cart
        Map<String, Object> cart = carts.get(cartId);
        if (cart == null) {
            cart = new HashMap<>();
            carts.put(cartId, cart);
        }

        // Add item to cart
        String itemKey = itemName + "_" + price;
        if (cart.containsKey(itemKey)) {
            // Item already exists, update quantity
            int existingQty = (int) cart.get(itemKey);
            cart.put(itemKey, existingQty + quantity);
        } else {
            cart.put(itemKey, quantity);
        }

        // Calculate total
        double total = 0;
        for (String key : cart.keySet()) {
            String[] parts = key.split("_");
            double itemPrice = Double.parseDouble(parts[1]);
            int itemQty = (int) cart.get(key);
            total = total + (itemPrice * itemQty);
        }

        System.out.println("Cart " + cartId + " total: " + total);

        return "Item added. Total: " + total;
    }

    @GetMapping("/getTotal")
    public String getTotal(@RequestParam("cartId") String cartId) {
        Map<String, Object> cart = carts.get(cartId);

        if (cart == null) {
            return "Cart not found";
        }

        // Calculate total
        double total = 0;
        for (String key : cart.keySet()) {
            String[] parts = key.split("_");
            double itemPrice = Double.parseDouble(parts[1]);
            int itemQty = (int) cart.get(key);
            total = total + (itemPrice * itemQty);
        }

        return "Total: " + total;
    }
}
```