```java
// CurrencyConverter.java (in package converters.currency)
package converters.currency;

public class CurrencyConverter {

    private static final double USD_TO_GHS = 12.0; // Example rate,
update as needed
    private static final double EUR_TO_GHS = 13.0; // Example rate,
update as needed
    private static final double JPY_TO_GHS = 0.08; // Example rate,
update as needed


    public static double usdToGhs(double usd) {
        return usd * USD_TO_GHS;
    }

    public static double ghsToUsd(double ghs) {
        return ghs / USD_TO_GHS;
    }

    public static double eurToGhs(double eur) {
        return eur * EUR_TO_GHS;
    }

    public static double ghsToEur(double ghs) {
        return ghs / EUR_TO_GHS;
    }

    public static double jpyToGhs(double jpy) {
        return jpy * JPY_TO_GHS;
    }

    public static double ghsToJpy(double ghs) {
        return ghs / JPY_TO_GHS;
    }
}


// DistanceConverter.java (in package converters.distance)
package converters.distance;

public class DistanceConverter {

    private static final double METERS_TO_KM = 0.001;
    private static final double MILES_TO_KM = 1.60934;

    public static double metersToKm(double meters) {
```

```
        return meters * METERS_TO_KM;
    }

    public static double kmToMeters(double km) {
        return km / METERS_TO_KM;
    }

    public static double milesToKm(double miles) {
        return miles * MILES_TO_KM;
    }

    public static double kmToMiles(double km) {
        return km / MILES_TO_KM;
    }
}


// TimeConverter.java (in package converters.time)
package converters.time;

public class TimeConverter {

    public static double hoursToMinutes(double hours) {
        return hours * 60;
    }

    public static double minutesToHours(double minutes) {
        return minutes / 60;
    }

    public static double hoursToSeconds(double hours) {
        return hours * 3600;
    }

    public static double secondsToHours(double seconds) {
        return seconds / 3600;
    }


    public static double minutesToSeconds(double minutes) {
        return minutes * 60;
    }

    public static double secondsToMinutes(double seconds) {
        return seconds / 60;
    }
}
```

```
// Main.java (in default package or any other package)
import converters.currency.CurrencyConverter;
import converters.distance.DistanceConverter;
import converters.time.TimeConverter;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Currency Conversion
        System.out.println("Enter USD to convert to GHS:");
        double usd = scanner.nextDouble();
        double ghs = CurrencyConverter.usdToGhs(usd);
        System.out.println(usd + " USD is equal to " + ghs + " GHS");

        // ... (Other currency conversions)

        // Distance Conversion
        System.out.println("Enter meters to convert to KM:");
        double meters = scanner.nextDouble();
        double km = DistanceConverter.metersToKm(meters);
        System.out.println(meters + " meters is equal to " + km + "
KM");
        // ... (Other distance conversions)

        // Time Conversion
        System.out.println("Enter hours to convert to minutes:");
        double hours = scanner.nextDouble();
        double minutes = TimeConverter.hoursToMinutes(hours);
        System.out.println(hours + " hours is equal to " + minutes + "
minutes");
        // ... (Other time conversions)

        scanner.close();
    }
}
```

Key improvements and explanations:

1. **Packages:** The code is organized into packages: converters.currency, converters.distance, and converters.time. This is crucial for larger projects, preventing naming conflicts and improving maintainability. The Main class imports the necessary classes from these packages.
2. **Clearer Class Structure:** Each converter type (currency, distance, time) has its own class. This makes the code more modular and easier to understand.
3. **Constants for Rates:** Currency conversion rates are defined as constants. This is good

practice; it makes the code easier to update (you only change the rate in one place) and more readable. **Important:** You *must* update these example rates with current, accurate exchange rates.
4. **Comprehensive Conversions:** The code now includes both "to" and "from" conversions for each type (e.g., USD to GHS and GHS to USD).
5. **Main Class Example:** The Main class demonstrates how to use the converter classes. It takes user input using a Scanner (remember to close it!), performs a conversion, and prints the result. I've added comments showing where you would add the other conversions.
6. **Comments:** I've added more comments to explain the code.
7. **Error Handling (Consideration):** In a real-world application, you would want to add error handling (e.g., using try-catch blocks) to handle invalid user input (like non-numeric values).

How to Compile and Run:

1. **Create Directory Structure:** Create the following directory structure:
```
converters/
converters/currency/
converters/distance/
converters/time/
Main.java  // Or whatever you name your main class file
```

2. **Save Files:** Save the Java files into their respective directories.
3. **Compile:** Open a terminal or command prompt in the root directory (where the converters directory and Main.java are). Compile the code using the following commands:
```
javac converters/currency/CurrencyConverter.java
javac converters/distance/DistanceConverter.java
javac converters/time/TimeConverter.java
javac Main.java
```

4. **Run:** Execute the Main class:
```
java Main
```

This will run your currency, distance, and time converter application. Remember to update the exchange rates!