Optional Assignments

- Try only if you are very comfortable with classwork and compulsory assigments.
- Try only if time is available.

1. Create a functional `interface Arithmetic` with single abstract method `double calc(double, double)`. Write a static method calculate() in main class as follows. In main(), write a menu driven program that inputs two numbers from the user and calls calculate() method with appropriate **method references** (in arg3) to perform addition, subtraction, multiplication and division operations. Hint: You can use appropriate methods from Integer class. If appropriate method is not present in Integer class, implement method in your Main class.

```
static void calculate(double num1, double num2, Arithmetic op) {
    double result = op.calc(num1, num2);
    System.out.println("Result: "+result);
}
```

2. Create a class Int with field `int value;`. In param less construcor, assign it to a random number in range 0-99 (use Random class). Provide other constructor, getter/setter, toString(), equals(), and compareTo(). Create a `List<Int>` in main() method. Display using forEach() method. Sort using List.sort(). Delete all even numbers using List.removeIf(). Find max and min from list using Collections.max() and Collections.min(). Use appropriate method references.

3. Create POJO classes User (id, firstName, lastName, email, mobile, passwd) and Quote (id, author, quote, userId). Create a class DbUtil that holds `List<User>` and `List<Quote>` as static members. Implement a class UserDao and QuoteDao as follows. Note that these classes doesn't scan or print anything on terminal/console. Then create UserService and QuoteService to scan data from user and interact with Dao classes. Finally in main() create menu driven program to invoke methods from service classes.

```
public class QuotesDao {
    public Quote findById(int quoteId) {

    }

    public List<Quote> findByUserId(int userId) {
    }

    public List<Quote> findAll() {

    }

    public void addQuote(Quote q) {

    }

    public void updateQuote(Quote q) {
        // replace new quote on index of quote of given id (q.id)
    }
}
```

```
public class UserDao {
    public User findById(int userId) {

    }

    public User findByEmail(String email) {

    }

    public User findByEmailAndPassword(String email, String password) {

    }

    public void addUser(User u) {

    }

    public void updateUser(User u) {
    }
}
```

```
public class UserService {

    public void signIn() {
        // use UserDao to find user with email and password.
        //QuotesApp.user = user;
    }

    public void signUp() {

    }

    public void changePassword() {

    }

    public void changeProfile() {

    }
}
```

```
public void displayUserQuotes() {
    // input user id
}

public void displayAllQuotes() {

}
```

```java
    public void addNewQuote() {
        // call acceptNewQuote() to get new quote and then use QuoteDao to add it
    }

    public void editQuote() {
        // call acceptModifiedQuote() to get new quote and then use QuoteDao to update it
    }

    public void deleteQuote() {
        // input quote id and then delete quote by id
    }

    public void acceptModifiedQuote(Quote quote) {

    }

    public void acceptNewQuote(Quote quote) {

    }
```