Exercise 26

```
class MyThread extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++)
            System.out.println("Running: " + i);
    }

    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();
        t1.start();
        t2.start();
    }
}
```

Exercise 27

```
import java.util.*;

class LambdaSort {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Zack", "Alex", "John");
        names.sort((a, b) -> a.compareTo(b));
        names.forEach(System.out::println);
    }
}
```

Exercise 28

```
import java.util.*;
import java.util.stream.*;

class StreamEven {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(10, 15, 20, 25, 30);
        nums.stream().filter(n -> n % 2 == 0).forEach(System.out::println);
    }
}
```

Exercise 29

```
record Person(String name, int age) {}

class RecordDemo {
    public static void main(String[] args) {
        List<Person> people = List.of(new Person("Alice", 22), new Person("Bob", 30));
        people.stream().filter(p -> p.age() >= 25).forEach(System.out::println);
    }
}
```

Exercise 30

```
class PatternSwitch {
    static void printType(Object obj) {
        switch (obj) {
            case Integer i -> System.out.println("Integer: " + i);
            case String s -> System.out.println("String: " + s);
            case Double d -> System.out.println("Double: " + d);
            default -> System.out.println("Unknown type");
        }
    }

    public static void main(String[] args) {
        printType(42);
        printType("Hello");
        printType(3.14);
    }
}
```

Exercise 31

```
import java.sql.*;

class JdbcSelect {
    public static void main(String[] args) throws Exception {
         Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"root", "password");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM students");
        while (rs.next()) {
            System.out.println(rs.getInt(1) + " " + rs.getString(2));
        }
        con.close();
    }
}
```

Exercise 32

```
class StudentDAO {
    public void insertStudent(Connection con, int id, String name) throws Exception {
          PreparedStatement ps = con.prepareStatement("INSERT INTO students VALUES (?,
?)");
        ps.setInt(1, id);
        ps.setString(2, name);
        ps.executeUpdate();
    }

    public void updateStudent(Connection con, int id, String name) throws Exception {
         PreparedStatement ps = con.prepareStatement("UPDATE students SET name=? WHERE
id=?");
        ps.setString(1, name);
        ps.setInt(2, id);
        ps.executeUpdate();
    }
}
```

Exercise 33

```
class TransactionExample {
     public static void transfer(Connection con, int fromId, int toId, double amount)
throws Exception {
        try {
            con.setAutoCommit(false);
             PreparedStatement debit = con.prepareStatement("UPDATE accounts SET balance
= balance - ? WHERE id = ?");
            debit.setDouble(1, amount);
            debit.setInt(2, fromId);
            debit.executeUpdate();

             PreparedStatement credit = con.prepareStatement("UPDATE accounts SET balance
= balance + ? WHERE id = ?");
            credit.setDouble(1, amount);
            credit.setInt(2, toId);
            credit.executeUpdate();

            con.commit();
            System.out.println("Transfer successful.");
        } catch (Exception e) {
            con.rollback();
            System.out.println("Transfer failed. Rolled back.");
        }
    }
}
```

Exercise 34

```
// module-info.java in com.utils
module com.utils {
    exports com.utils;
}

// module-info.java in com.greetings
module com.greetings {
    requires com.utils;
}
```

Exercise 35

```
// Server
import java.net.*;
import java.io.*;

class ChatServer {
    public static void main(String[] args) throws Exception {
        ServerSocket server = new ServerSocket(1234);
        Socket socket = server.accept();
                                    BufferedReader    in    =    new    BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        out.println("Hello Client");
        System.out.println("Client says: " + in.readLine());
        socket.close();
        server.close();
    }
}

// Client
class ChatClient {
    public static void main(String[] args) throws Exception {
        Socket socket = new Socket("localhost", 1234);
                                    BufferedReader    in    =    new    BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        System.out.println("Server says: " + in.readLine());
        out.println("Hello Server");
        socket.close();
    }
}
```

Exercise 36

```
import java.net.URI;
import java.net.http.*;

class HttpExample {
    public static void main(String[] args) throws Exception {
        HttpClient client = HttpClient.newHttpClient();
                                                    HttpRequest       request       =
HttpRequest.newBuilder().uri(URI.create("https://api.github.com")).build();
                            HttpResponse<String>    response    =    client.send(request,
HttpResponse.BodyHandlers.ofString());
        System.out.println("Status: " + response.statusCode());
        System.out.println(response.body());
    }
}
```

Exercise 37

```
class Sample {
    public void greet() {
        System.out.println("Hello");
    }
}
// Compile and run: javap -c Sample
```

Exercise 38

```
// Use JD-GUI or CFR to decompile compiled Java class files.
// 1. Compile a .java file
// 2. Open the .class file using JD-GUI
```

Exercise 39

```
class ReflectionDemo {
    public void sayHello() {
        System.out.println("Hello from reflection!");
    }

    public static void main(String[] args) throws Exception {
        Class<?> cls = Class.forName("ReflectionDemo");
        Object obj = cls.getDeclaredConstructor().newInstance();
        cls.getDeclaredMethod("sayHello").invoke(obj);
    }
}
```

Exercise 40

```
class VirtualThreads {
    public static void main(String[] args) {
        for (int i = 0; i < 100; i++) {
                    Thread.startVirtualThread(() -> System.out.println("Running virtual
thread"));
        }
    }
}
```

Exercise 41

```
import java.util.concurrent.*;

class CallableExample {
    public static void main(String[] args) throws Exception {
        ExecutorService executor = Executors.newFixedThreadPool(2);
        Callable<String> task = () -> "Hello from Callable!";
        Future<String> future = executor.submit(task);
        System.out.println("Result: " + future.get());
        executor.shutdown();
    }
}
```