

Info

- All programs use a package **rainbow** written by me to show colored and formatted output.

Question1

Implementing a time counter via a separate thread, and call the function whenever user asks to show the time.

Code

```
import java.util.Scanner;
import rainbow.rainbow;

class ClockWorker extends Thread {
    private int counter = 0;
    private final String context = "[ " + rainbow.green(this.getName()) + "
] ";

    ClockWorker() {
        System.out.println(this.context + "ClockWorker started");
    }

    @Override
    public void run() {
        try {
            while (true) {
                this.counter += 1;
                Thread.sleep(1 * 1000);
            }

        } catch (InterruptedException e) {
            System.out.println(context + "Exception !");
        }
    }

    public int getSeconds() {
        System.out.print(context);
        return this.counter;
    }
}

public class Clock {

    public static void main(String args[]) {
        String command;
        final Scanner handler = new Scanner(System.in);
        final String context = "[ " +
rainbow.green(Thread.currentThread().getName()) + " ] ";
```

```
// tell user about current thread and that counter has started
System.out.println(context + " Time Counter started");

// start separate thread with timer
final ClockWorker worker = new ClockWorker();
worker.start();

while (true) {
    System.out.print(context + "Enter \"show\" to show live timer :
");
    command = handler.nextLine();
    if (command.equals("show")) {
        System.out.println("Seconds elapsed : " +
worker.getSeconds());
    }
}
}
```

Output

```
yash@hephaestus: ~  
java -cp ./ Clock  
→ question1 git:(19BCE2669) X javac rainbow/*.java Clock.java  
→ question1 git:(19BCE2669) X java -cp ./ Clock  
[ main ] Time Counter started  
[ Thread-0 ] ClockWorker started  
[ main ] Enter "show" to show live timer :   
    try {  
        while (true) {  
            this.counter += 1  
            Thread.sleep(1 * 1000);  
        }  
    }  
    } catch (InterruptedException e) {  
        System.out.println(context + "Exception !");  
    }  
}
```

```
yash@hephaestus: ~  
java -cp ./ Clock  
→ question1 git:(19BCE2669) X javac rainbow/*.java Clock.java  
→ question1 git:(19BCE2669) X java -cp ./ Clock  
[ main ] Time Counter started  
[ Thread-0 ] ClockWorker started  
[ main ] Enter "show" to show live timer : show  
[ Thread-0 ] Seconds elapsed : 41  
[ main ] Enter "show" to show live timer :   

```

```

yash@hephaestus: ~
java -cp ./ Clock
→ question1 git:(19BCE2669) X javac rainbow/*.java Clock.java
→ question1 git:(19BCE2669) X java -cp ./ Clock
[ main ] Time Counter started
[ Thread-0 ] ClockWorker started
[ main ] Enter "show" to show live timer : show
[ Thread-0 ] Seconds elapsed : 41
[ main ] Enter "show" to show live timer : show
[ Thread-0 ] Seconds elapsed : 56
[ main ] Enter "show" to show live timer :

yash@hephaestus: ~/Desktop/files/works/projects/playing-with-java/final-da/threading/question1
→ question1 git:(19BCE2669) X javac rainbow/*.java Clock.java
→ question1 git:(19BCE2669) X java -cp ./ Clock
[ main ] Time Counter started
[ Thread-0 ] ClockWorker started
[ main ] Enter "show" to show live timer : show
[ Thread-0 ] Seconds elapsed : 41
[ main ] Enter "show" to show live timer : show
[ Thread-0 ] Seconds elapsed : 56
[ main ] Enter "show" to show live timer : show
[ Thread-0 ] Seconds elapsed : 65
[ main ] Enter "show" to show live timer : ^C
Execution time: 0h:01m:06s sec
→ question1 git:(19BCE2669) X

```

Question2

Implementing two threads to calculate the nth prime number and prime factors of a number and also show the current threads being executed.

Code

```

import java.util.Scanner;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;
import rainbow.rainbow;
import rainbow.rainbow;

class PrimeNumber extends Thread {
    private int nth;
    private final String context = "[" + rainbow.green(this.getName()) + "
] ";

```

```
public PrimeNumber(int nth) {
    this.nth = nth;
}

@Override
public void run() {
    System.out.println(this.context + this.nth + "th prime number is "
+ this.findNth(this.nth));
}

private int findNth(int nth) {
    int num, count, i;
    num = 1;
    count = 0;

    while (count < nth) {
        num = num + 1;
        for (i = 2; i <= num; i++) {
            if (num % i == 0) {
                break;
            }
        }
        if (i == num) {
            count = count + 1;
        }
    }

    return num;
}

}

class PrimeFactors extends Thread {
    private int number;
    private Integer[] factors;
    private final String context = "[ " + rainbow.green(this.getName()) + "
] ";

    public PrimeFactors(int number) {
        this.number = number;
    }

    @Override
    public void run() {
        System.out.print(this.context + "Prime Factors of " + this.number +
" are : ");
        this.calculator();
        for (Integer num : this.factors) {
            System.out.print(num + " ");
        }
        System.out.println();
    }

    private void calculator() {
        List<Integer> list = new ArrayList<Integer>();
```

```
        for (int i = 2; i < this.number; i++) {
            while (this.number % i == 0) {
                list.add(i);
                this.number = this.number / i;
            }
        }
        this.factors = list.toArray(new Integer[list.size()]);
    }
}

public class Computations {
    public static void main(String args[]) {
        final String context = "[ " +
rainbow.green(Thread.currentThread().getName()) + " ] ";

        try {
            Scanner handler = new Scanner(System.in);
            System.out.print(context + rainbow.bold("Enter a number : "));
            int number = handler.nextInt();

            PrimeNumber primeNumber = new PrimeNumber(number);
            primeNumber.start();

            PrimeFactors primeFactors = new PrimeFactors(number);
            primeFactors.start();
        } catch (IllegalArgumentException e) {
            System.out.println(rainbow.red(context + "Illegal arguments
!"));
        } catch (InputMismatchException e) {
            System.out.println(rainbow.red(context + "Invalid input type
!"));
        } catch (Exception e) {
            System.out.println(rainbow.red(context + "Some Error
encountered !"));
        }
    }
}
```

Output

```

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question2
→ question2 git:(19BCE2669) X java -cp ./ Computations
[ main ] Enter a number : 250
[ Thread-1 ] Prime Factors of 250 are : 2 5 5 5
[ Thread-0 ] 250th prime number is 1583
→ question2 git:(19BCE2669) X java -cp ./ Computations
[ main ] Enter a number : 400
[ Thread-1 ] Prime Factors of 400 are : 2 2 2 2 5 5
[ Thread-0 ] 400th prime number is 2741
→ question2 git:(19BCE2669) X java -cp ./ Computations
[ main ] Enter a number : 10000
[ Thread-1 ] Prime Factors of 10000 are : 2 2 2 2 5 5 5 5
[ Thread-0 ] 10000th prime number is 104729
Execution time: 0h:00m:04s sec
→ question2 git:(19BCE2669) X

```

Question3

Create two threads. One to make a web request and download the html of a page, and another to write the data to file.

Code

```

import java.net.URI;
import java.util.Scanner;
import java.io.FileWriter;
import java.io.IOException;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import rainbow.rainbow;

class FileWriterThread extends Thread {
    private String data;
    private String fileName;
    private FileWriter fileWriter = null;
    private final String context = "[ " + rainbow.green(this.getName()) + "
] ";

    public FileWriterThread(String fileName, String data) throws
IOException {
        System.out.println(this.context + "File Writer Thread
initialized");
        this.data = data;
        this.fileName = fileName;
        this.fileWriter = new FileWriter("./" + fileName);
    }

```

```

        @Override
        public void run() {
            try {
                this.fileWriter.write(this.data);
                System.out.println(this.context + "Data written to file : " +
this.fileName);
            } catch (IOException e) {
                System.out.print(e);
            }
        }

        @Override
        protected void finalize() throws Throwable {
            System.out.println(this.context + "File Writer closed.");
            this.fileWriter.close();
        }
    }

    class FileDownloader extends Thread {
        private String fileUrl;
        private String fileName;
        private final String context = "[ " + rainbow.green(this.getName()) + "
] ";

        public FileDownloader(String fileUrl, String fileName) {
            System.out.println(this.context + "File Downloader Thread
initialized");
            this.fileUrl = fileUrl;
            this.fileName = fileName;
        }

        public void run() {
            try {
                final HttpClient client = HttpClient.newHttpClient();
                final HttpRequest request =
HttpRequest.newBuilder().uri(URI.create(this.fileUrl)).build();
                final HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
                final String dataToWriteToFile = response.body();
                final FileWriterThread fileWriterThread = new
FileWriterThread(this.fileName, dataToWriteToFile);
                fileWriterThread.start();

                } catch (Exception e) {
                    System.out.println(this.context + "Error downloading webpage");
                }
            }
        }

        public class FileWorker {

            public static void main(final String[] args) {
                final Scanner handler = new Scanner(System.in);
                final String context = "[ " +

```



```
rainbow.green(Thread.currentThread().getName()) + " ] ";

    // take input of url
    System.out.print(context + rainbow.italic("Enter url of the page
you want to save : "));
    String url = handler.nextLine();

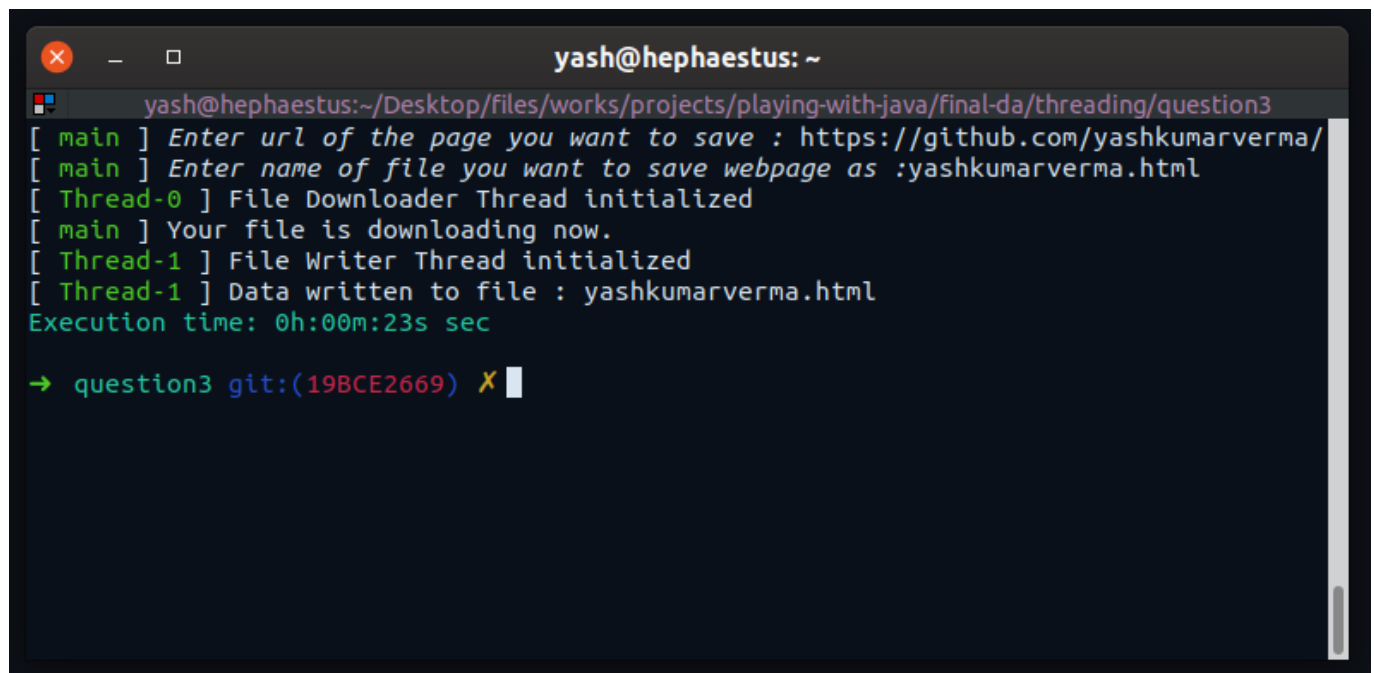
    // take input of filename
    System.out.print(context + rainbow.italic("Enter name of file you
want to save webpage as :"));
    String fileName = handler.nextLine();

    // download file
    FileDownloader fileDownloader = new FileDownloader(url, fileName);
    fileDownloader.start();

    System.out.println(context + "Your file is downloading now.");

    handler.close();
}
}
```

Output



```
yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question3
[ main ] Enter url of the page you want to save : https://github.com/yashkumarverma/
[ main ] Enter name of file you want to save webpage as :yashkumarverma.html
[ Thread-0 ] File Downloader Thread initialized
[ main ] Your file is downloading now.
[ Thread-1 ] File Writer Thread initialized
[ Thread-1 ] Data written to file : yashkumarverma.html
Execution time: 0h:00m:23s sec
→ question3 git:(19BCE2669) X
```

Content of file : "yashkumarverma.html"

```

File: yashkumarverna.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <link rel="dns-prefetch" href="https://github.githubassets.com">
    <link rel="dns-prefetch" href="https://avatars0.githubusercontent.com">
    <link rel="dns-prefetch" href="https://avatars1.githubusercontent.com">
    <link rel="dns-prefetch" href="https://avatars2.githubusercontent.com">
    <link rel="dns-prefetch" href="https://avatars3.githubusercontent.com">
    <link rel="dns-prefetch" href="https://github-cloud.s3.amazonaws.com">
    <link rel="dns-prefetch" href="https://user-images.githubusercontent.com/">
    <link crossorigin="anonymous" media="all" integrity="sha512-QV1ZN8jZz8stPx+uh4ZAKc6AJ1z8A9VHut/SGtgbc+iYLfhrh68QmDH3rZgkXJ0BWI0cDw+ILnWcctH0ljcHPg==" rel="stylesheet" href="https://github.githubassets.com/assets/frameworks-415d593418d9cfc2d3f1fae87864029.css" />
    <link crossorigin="anonymous" media="all" integrity="sha512-Z1L3RiXbv3iU15ESvB9fNSNb9jKTywFzKK2vJTr9WGTx+CkdM33cn+TfTEAzgI9LgVi8sz2Ej12Q6jm07BxQ==" rel="stylesheet" href="https://github.githubassets.com/assets/site-6759774625dbbf1de2535e444af07d7c.css" />
    <link crossorigin="anonymous" media="all" integrity="sha512-o6YKGhQ+KL5BHp09YZle7hHKStubbXI+7RXqznp5YSpmWNW+IAJlGcyQfgfYhvjqJ7GugCjRB91j6qchficWXg==" rel="stylesheet" href="https://github.githubassets.com/assets/github-a3a60a1a143e2a5e411e9a3d61995ee.css" />
    <script crossorigin="anonymous" defer="defer" integrity="sha512-g4ztuyuFPzjTviQYBeZdHEDaHz2K6RCz4RsZsnL3m5ko4kiWCjB9W6uIScLkNr8L/BtC2dYLI Fk0dOLDYBHLQ==" type="application/javascript" data-module-id="/compat.js" data-src="https://github.githubassets.com/assets/compat-838cedbb.js"></script>
    <script crossorigin="anonymous" defer="defer" integrity="sha512-dCwmfQLwCHY085Txw89N+CudKaNZ9QhDIHrNFUPFYMqr8EFrRHFp/Bico7AEoObA/pgR1JM9AeagIJnXpT1A==" type="application/javascript" src="https://github.githubassets.com/assets/environment-742c267e.js"></script>
    <script crossorigin="anonymous" defer="defer" integrity="sha512-S2gekJF0IdKEqb3PK+eMAWHP0P2sx8mQ+6aBSGArCvnm+pMnHOuynzi4NNm6P41mI1qES7cQ41VF4vQ5jJG+A==" type="application/javascript" src="https://github.githubassets.com/assets/chunk-frameworks-4b681e90.js"></script>
    <script crossorigin="anonymous" defer="defer" integrity="sha512-Shix6HKvSDNREARBi1dz6K2DezL6x88FzVNZPUHKBjUk6iZAUHacGedCwe/1YQtFhf8DzLIIDhr1sb41uwlIXg==" type="application/javascript" src="https://github.githubassets.com/assets/chunk-vendor-4a18b1e8.js"></script>

```

Question4

Write a JAVA Program to create parallel connections to database using JDBC driver, and insert data into database via different threads.

Code

```

/** libraries for database */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import rainbow.rainbow;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

/** class to handle all database operations */
class DatabaseWorker extends Thread {
    private final String url = "jdbc:mysql://localhost:3306/java_db";
    private final String user = "yash";
    private final String password = "yash2000.";
    private Connection connection = null;
    private Statement statement = null;
    private final String context = "[ " + rainbow.green(this.getName()) + "
] ";

```

```
public void run() {
    System.out.println(context + "database connected");
}

/** constructor to create a connection and store */
public DatabaseWorker() {
    try {
        System.out.println(this.context + "Connecting to database");
        this.connection = DriverManager.getConnection(this.url,
this.user, this.password);
        this.statement = connection.createStatement();
        System.out.println(this.context + "Database connected");

    } catch (Exception e) {
        System.out.println(this.context + "Error connecting to
database;");
    }
}

/** function to insert a new user into database */
public boolean insertUser(String name, String regNo, String mobile, int
age) {
    try {
        final String sqlQuery = "INSERT INTO users VALUES ('" + name +
"', '" + regNo + "', '" + mobile + "', '" + age
+ "');"
        System.out.println(this.context + "sql > " + sqlQuery);
        this.statement.executeUpdate(sqlQuery);
        return true;
    } catch (SQLException e) {
        System.out.println(this.context + "Error while inserting entry
to database");
        System.out.println(e.getErrorCode());
        return false;
    } catch (Exception e) {
        System.out.println(this.context + "Unhandled Exception !");
        return false;
    }
}

}

/** main worker class */
public class ParallelDatabase {

    public static void main(String args[]) throws Exception {
        final String context = "[ " +
rainbow.green(Thread.currentThread().getName()) + " ] ";
        ExecutorService executor = Executors.newFixedThreadPool(10);

        // create 10 parallel connections and insert data
        for (int i = 0; i < 10; i++) {
            Runnable worker = new MyRunnable(i);
            executor.execute(worker);
        }
    }
}
```

```

        // wait for all operations
        executor.shutdown();
        while (!executor.isTerminated()) {
        }
        System.out.println(context + "Finished Writing.");
    }

    public static class MyRunnable implements Runnable {
        private final int index;
        private DatabaseWorker worker = new DatabaseWorker();

        MyRunnable(int index) {
            this.index = index;
        }

        @Override
        public void run() {
            this.worker.insertUser("user_" + this.index, "19BCE00" +
            this.index, "pass@" + this.index,
                (20 + this.index) % 30);
        }
    }
}

```

Output

Commands executed by each thread have the thread name towards the left hand side in the format of **Thread-{threadNumber}**

```

→ question4 git:(19BCE2669) X javac --module-path com/mysql/jdbc -cp . ParallelDatabase.java
Execution time: 0h:00m:03s sec
→ question4 git:(19BCE2669) X java -cp ./ ParallelDatabase
[ Thread-0 ] Connecting to database
[ Thread-0 ] Database connected
[ Thread-1 ] Connecting to database
[ Thread-1 ] Database connected
[ Thread-2 ] Connecting to database
[ Thread-2 ] Database connected
[ Thread-3 ] Connecting to database
[ Thread-3 ] Database connected
[ Thread-4 ] Connecting to database
[ Thread-4 ] Database connected
[ Thread-2 ] sql > INSERT INTO users VALUES ('user_2','19BCE002', 'pass@2',22);
[ Thread-3 ] sql > INSERT INTO users VALUES ('user_3','19BCE003', 'pass@3',23);
[ Thread-1 ] sql > INSERT INTO users VALUES ('user_1','19BCE001', 'pass@1',21);
[ Thread-4 ] sql > INSERT INTO users VALUES ('user_4','19BCE004', 'pass@4',24);
[ Thread-0 ] sql > INSERT INTO users VALUES ('user_0','19BCE000', 'pass@0',20);
[ Thread-5 ] Connecting to database
[ Thread-5 ] Database connected
[ Thread-6 ] Connecting to database
[ Thread-6 ] Database connected
[ Thread-5 ] sql > INSERT INTO users VALUES ('user_5','19BCE005', 'pass@5',25);
[ Thread-6 ] sql > INSERT INTO users VALUES ('user_6','19BCE006', 'pass@6',26);
[ Thread-7 ] Connecting to database
[ Thread-7 ] Database connected
[ Thread-8 ] Connecting to database
[ Thread-8 ] Database connected
[ Thread-7 ] sql > INSERT INTO users VALUES ('user_7','19BCE007', 'pass@7',27);
[ Thread-9 ] Connecting to database
[ Thread-9 ] Database connected
[ Thread-8 ] sql > INSERT INTO users VALUES ('user_8','19BCE008', 'pass@8',28);
[ Thread-9 ] sql > INSERT INTO users VALUES ('user_9','19BCE009', 'pass@9',29);
[ main ] Finished Writing.
Execution time: 0h:00m:03s sec
→ question4 git:(19BCE2669) X

```

Question5

Write a JAVA program to simulate a client server architecture with server occupying a thread and each client being a different thread. Send messages from clients to server at regular intervals of time.

Code

```
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;
import rainbow.rainbow;

class Server extends Thread {
    private final String context = "[ " + rainbow.green(this.getName()) + "
] ";
    public static BlockingQueue<String> messages = new
LinkedBlockingQueue<String>();

    public Server() {
        System.out.println(this.context + " Server initialized");
    }

    public void run() {
        while (true) {
            String message;
            while ((message = Server.messages.poll()) != null) {
                System.out.println(this.context + "got message from " +
message);
            }
        }
    }
}

class Client extends Thread {
    private String context;
    private String name;
    private int coolDownTime;

    public Client(String name, int coolDownTime) {
        this.name = name;
        this.coolDownTime = coolDownTime;
        this.context = "[ " + rainbow.green(this.getName()) + " ] " +
rainbow
        .dim(rainbow.italic(" [ " + this.name + " pinging every " +
this.coolDownTime / 1000 + " seconds ] "));
    }

    public void run() {
        int messageNumber = 1;
        while (true) {
            try {
                Server.messages.put(this.context + " message #" +
messageNumber);
                messageNumber++;
                this.sleep(this.coolDownTime);
            }
        }
    }
}
```

```
        } catch (InterruptedException e) {
            System.out.println(this.context + "Error Sending message");
        }
    }
}

public class Communication {
    public static void main(String args[]) {
        Server server = new Server();
        server.start();

        Client client1 = new Client("client:1", 1 * 2 * 1000);
        Client client2 = new Client("client:2", 2 * 2 * 1000);
        Client client3 = new Client("client:3", 3 * 2 * 1000);
        Client client4 = new Client("client:4", 4 * 2 * 1000);

        // start all clients
        client1.start();
        client2.start();
        client3.start();
        client4.start();
    }
}
```

Output

```

→ question5 git:(19BCE2669) X javac rainbow/*.java Communication.java
→ question5 git:(19BCE2669) X java -cp . Communication
[ Thread-0 ] Server initialized
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #1
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #1
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #1
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #1
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #2
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #2
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #3
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #2
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #4
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #3
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #2
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #5
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #6
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #4
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #3
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #7
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #8
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #3
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #5
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #9

```



```

→ question5 git:(19BCE2669) X javac rainbow/*.java Communication.java
→ question5 git:(19BCE2669) X java -cp . Communication
[ Thread-0 ] Server initialized
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #1
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #1
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #1
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #1
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #2
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #2
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #3
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #2
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #4
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #3
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #2
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #5
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #6
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #4
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #3
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #7
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #8
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #3
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #5
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #9
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #4
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #10
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #6
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #11
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #12
[ Thread-0 ] got message from [ Thread-4 ] [ client:4 pinging every 8 seconds ] message #4
[ Thread-0 ] got message from [ Thread-3 ] [ client:3 pinging every 6 seconds ] message #5
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #7
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #13
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #14
[ Thread-0 ] got message from [ Thread-2 ] [ client:2 pinging every 4 seconds ] message #8
[ Thread-0 ] got message from [ Thread-1 ] [ client:1 pinging every 2 seconds ] message #15

```

Question6

Code

```

import rainbow.rainbow;

public class Worker {
    private int counter = 0;

    // synchronized keyword to allow only one thread to execute at one time
    public synchronized void increment(String threadName) throws
    InterruptedException {
        counter++;
        Thread.sleep(1000);
        System.out.println("Thread Working: " + threadName + " and counter
is: " + counter);
    }

    // function to invoke clients
    public void doWork() {

        // start first thread
        Thread thread1 = new Thread(new Runnable() {
            public void run() {

```



```
        for (int i = 0; i < 10; i++) {
            try {
                increment("[ " +
rainbow.green(Thread.currentThread().getName()) + " ] ");
            } catch (InterruptedException ex) {
                System.out.println("Error !");
            }
        }
    }
});
thread1.start();

// start second thread
Thread thread2 = new Thread(new Runnable() {
    public void run() {
        for (int i = 0; i < 10; i++) {
            try {
                increment("[ " +
rainbow.green(Thread.currentThread().getName()) + " ] ");
            } catch (InterruptedException ex) {
                System.out.println("Error !");
            }
        }
    }
});
thread2.start();

try {
    thread1.join();
    thread2.join();
} catch (InterruptedException ignored) {
    System.out.println(rainbow.red("Couldn't join threads"));
}
System.out.println("counter is: " + counter);
}

public static void main(String[] args) {
    Worker worker = new Worker();
    worker.doWork();
}
}
```

Output

When we don't use synchronized keyword, the threads execute the function at the same time, and we see undesirable results.

```

→ question6 git:(19BCE2669) X javac rainbow/*.java Worker.java
→ question6 git:(19BCE2669) X java -cp . Worker
counter is: 0
Thread Working: [ Thread-1 ] and counter is: 2
Thread Working: [ Thread-0 ] and counter is: 2
Thread Working: [ Thread-1 ] and counter is: 4
Thread Working: [ Thread-0 ] and counter is: 4
Thread Working: [ Thread-1 ] and counter is: 6
Thread Working: [ Thread-0 ] and counter is: 7
Thread Working: [ Thread-1 ] and counter is: 8
Thread Working: [ Thread-0 ] and counter is: 8
Thread Working: [ Thread-0 ] and counter is: 10
Thread Working: [ Thread-1 ] and counter is: 10
Thread Working: [ Thread-0 ] and counter is: 12
Thread Working: [ Thread-1 ] and counter is: 12
Thread Working: [ Thread-0 ] and counter is: 14
Thread Working: [ Thread-1 ] and counter is: 14
Thread Working: [ Thread-0 ] and counter is: 16
Thread Working: [ Thread-1 ] and counter is: 16
Thread Working: [ Thread-0 ] and counter is: 18
Thread Working: [ Thread-1 ] and counter is: 18
Thread Working: [ Thread-0 ] and counter is: 20
Thread Working: [ Thread-1 ] and counter is: 20
Execution time: 0h:00m:10s sec
→ question6 git:(19BCE2669) X

```

Question7

Write a JAVA Program to show usage of custom - user defined exceptions and override the base methods to change the way error messages are shown to user.

Code

```

import java.util.InputMismatchException;
import java.util.Scanner;
import rainbow.rainbow;

class InvalidNameException extends Exception {
    private final String context = "[ " +
rainbow.green("InvalidNameException") + " ] ";
    private String message;

    public InvalidNameException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.context + "triggered");
    }

    public String getMessage() {
        return this.context + this.message;
    }
}

```

```
};

class InvalidAgeException extends Exception {
    private final String context = "[ " +
rainbow.green("InvalidAgeException") + " ] ";
    private String message;

    public InvalidAgeException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.context + "triggered");
    }

    public String getMessage() {
        return this.context + this.message;
    }
};

class InvalidRegistrationNumberException extends Exception {
    private final String context = "[ " +
rainbow.green("InvalidRegistrationNumberException") + " ] ";
    private String message;

    public InvalidRegistrationNumberException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.context + "triggered");
    }

    public String getMessage() {
        return this.context + this.message;
    }
};

public class App {
    public static void main(String args[]) {
        final String context = "[ " + rainbow.green("App") + " ] ";

        try {
            Scanner handler = new Scanner(System.in);
            System.out.print(context + "Enter name : ");
            String name = handler.nextLine();
            if (name.length() <= 3) {
                throw new InvalidNameException("Name should be longer than
3 characters");
            }

            System.out.print(context + "Enter age : ");
            int age = handler.nextInt();
            handler.nextLine();
            if (age < 0 || age > 120) {
                throw new InvalidAgeException("Age should be positive and
less than 120");
            }
        }
    }
}
```

```

        System.out.print(context + "Enter registration number : ");
        String regNo = handler.nextLine();
        if (regNo.length() != 9) {
            throw new InvalidRegistrationNumberException("Registration
Number should be of 9 characters");
        }

        System.out.println(context + "Record Saved");

    } catch (InvalidNameException e) {
        System.out.println(e.getMessage());
    } catch (InvalidAgeException e) {
        System.out.println(e.getMessage());
    } catch (InvalidRegistrationNumberException e) {
        System.out.println(e.getMessage());
    } catch (Exception e) {
        System.out.println("Incorrect Data type entered");
    }
}
}

```

Output

```

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question7
→ question7 git:(19BCE2669) X java -cp . App
[ App ] Enter name : aa
[ InvalidNameException ] triggered
[ InvalidNameException ] Name should be longer than 3 characters
Execution time: 0h:00m:05s sec
→ question7 git:(19BCE2669) X

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question7
→ question7 git:(19BCE2669) X java -cp . App
[ App ] Enter name : yash verma
[ App ] Enter age : -2
[ InvalidAgeException ] triggered
[ InvalidAgeException ] Age should be positive and less than 120
Execution time: 0h:00m:05s sec
→ question7 git:(19BCE2669) X

```

```

yash@hephaestus: ~/Desktop/files/works/projects/playing-with-java/final-da/threading/question7
→ question7 git:(19BCE2669) X java -cp . App
[ App ] Enter name : yash verma
[ App ] Enter age : 20
[ App ] Enter registration number : 19bce
[ InvalidRegistrationNumberException ] triggered
[ InvalidRegistrationNumberException ] Registration Number should be of 9 characters
Execution time: 0h:00m:06s sec
→ question7 git:(19BCE2669) X

yash@hephaestus: ~/Desktop/files/works/projects/playing-with-java/final-da/threading/question7
→ question7 git:(19BCE2669) X java -cp . App
[ App ] Enter name : yash verma
[ App ] Enter age : 20
[ App ] Enter registration number : 19bce2669
[ App ] Record Saved
Execution time: 0h:00m:11s sec
→ question7 git:(19BCE2669) X

```

Question8

Write a JAVA program to show the usage of throws keyword with user defined exceptions

Code

```

import java.util.InputMismatchException;
import java.util.Scanner;
import rainbow.rainbow;

class InvalidNameException extends Exception {
    private final String context = "[ " +
rainbow.green("InvalidNameException") + " ] ";
    private String message;

    public InvalidNameException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.context + "triggered");
    }

    public String getMessage() {
        return this.context + this.message;
    }
};

class InvalidAgeException extends Exception {
    private final String context = "[ " +

```

```
rainbow.green("InvalidAgeException") + " ] ";
    private String message;

    public InvalidAgeException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.context + "triggered");
    }

    public String getMessage() {
        return this.context + this.message;
    }
};

class InvalidRegistrationNumberException extends Exception {
    private final String context = "[ " +
rainbow.green("InvalidRegistrationNumberException") + " ] ";
    private String message;

    public InvalidRegistrationNumberException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.context + "triggered");
    }

    public String getMessage() {
        return this.context + this.message;
    }
};

class User {
    final private String context = "[ " + rainbow.green("User") + " ] ";
    private String name;
    private String regNo;
    private int age;
    private Scanner handler = new Scanner(System.in);

    User() {
        System.out.println(this.context + "user instance created");
    }

    public void displayDetails() {
        System.out.println(this.context + "Name :" + this.name);
        System.out.println(this.context + "Age :" + this.age);
        System.out.println(this.context + "RegNo :" + this.regNo);
    }

    public void setName() throws InvalidNameException {
        System.out.print(context + "Enter name : ");
        this.name = handler.nextLine();
        if (name.length() <= 3) {
            throw new InvalidNameException("Name should be longer than 3
characters");
        }
    }
}
```

```
}

    public void setRegNo() throws InvalidRegistrationNumberException {
        System.out.print(context + "Enter registration number : ");
        this.regNo = handler.nextLine();
        if (regNo.length() != 9) {
            throw new InvalidRegistrationNumberException("Registration
Number should be of 9 characters");
        }
    }

    public void setAge() throws InvalidAgeException {
        System.out.print(context + "Enter age : ");
        this.age = handler.nextInt();
        handler.nextLine();
        if (age < 0 || age > 120) {
            throw new InvalidAgeException("Age should be positive and less
than 120");
        }
    }
}

public class App {
    public static void main(String args[]) {
        try {
            User user = new User();
            user.setName();
            user.setAge();
            user.setRegNo();
            user.displayDetails();
        } catch (InvalidNameException e) {
            System.out.println(e.getMessage());
        } catch (InvalidAgeException e) {
            System.out.println(e.getMessage());
        } catch (InvalidRegistrationNumberException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println("Incorrect Data type entered");
        }
    }
}
```

Output


```

throw new InvalidRegistrationNumberException("Registration Number should be of

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question8
→ question8 git:(19BCE2669) X javac rainbow/*.java App.java
→ question8 git:(19BCE2669) X java -cp . App
[ User ] user instance created
[ User ] Enter name : aa
[ InvalidNameException ] triggered Enter age : "
[ InvalidNameException ] Name should be longer than 3 characters
→ question8 git:(19BCE2669) X

```

```

handler.nextLine();

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question8
→ question8 git:(19BCE2669) X java -cp . App
[ User ] user instance created
[ User ] Enter name : yash verma
[ User ] Enter age : 150
[ InvalidAgeException ] triggered
[ InvalidAgeException ] Age should be positive and less than 120
Execution time: 0h:00m:06s sec
→ question8 git:(19BCE2669) X

```

```

public void setAge() throws InvalidAgeException {

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question8
→ question8 git:(19BCE2669) X java -cp . App
[ User ] user instance created
[ User ] Enter name : yash
[ User ] Enter age : 20
[ User ] Enter registration number : 2020202020202
[ InvalidRegistrationNumberException ] triggered
[ InvalidRegistrationNumberException ] Registration Number should be of 9 characters
Execution time: 0h:00m:06s sec
→ question8 git:(19BCE2669) X

```

```

this.age = handler.nextInt();

yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/playing-with-java/final-da/threading/question8
→ question8 git:(19BCE2669) X java -cp . App
[ User ] user instance created
[ User ] Enter name : yash
[ User ] Enter age : 20
[ User ] Enter registration number : 19BCE2669
[ User ] Name :yash
[ User ] Age :20
[ User ] RegNo :19BCE2669
Execution time: 0h:00m:06s sec
→ question8 git:(19BCE2669) X

```