Practical-5

Aim:-

Define a class representing a vehicle with properties like make, model, and year. Implement methods to display the vehicle details and calculate the mileage.

Create child classes like Car and Motorcycle that inherit from the Vehicle class and add specific properties and methods.

Code:-

```
class Vehicle {
  constructor(make, model, year) {
    this.make = make;
    this.model = model;
    this.year = year;
  }

  displayDetails() {
    console.log(`Vehicle: ${this.year} ${this.make} ${this.model}`);
  }

  calculateMileage(distance, fuel) {
    const mileage = distance / fuel;
    console.log(`Mileage: ${mileage} miles per liter`);
  }
}

class Car extends Vehicle {
  constructor(make, model, year, doors) {
    super(make, model, year);
    this.doors = doors;
  }

  displayDetails() {
    super.displayDetails();
    console.log(`Doors: ${this.doors}`);
  }
}

class Motorcycle extends Vehicle {
  constructor(make, model, year, engineDisplacement) {
    super(make, model, year);
    this.engineDisplacement = engineDisplacement;
  }
```

```
  displayDetails() {
    super.displayDetails();
    console.log(`Engine Displacement: ${this.engineDisplacement} cc`);
  }
}

const car = new Car("Supra", 2022, 2, 2);
car.displayDetails();
car.calculateMileage(300, 15);

const motorcycle = new Motorcycle("Ninja", 2021, 500);
motorcycle.displayDetails();
motorcycle.calculateMileage(200, 10);
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr5.js"
Vehicle: 2 Supra 2022
Doors: 2
Mileage: 20 miles per liter
Vehicle: 500 Ninja 2021
Engine Displacement: undefined cc
Mileage: 20 miles per liter
PS E:\Darsh\AWT>
```

Practical-6

Aim:-

Use the prototype property to add a new method to an existing object constructor, such as Array or String.

Code:-

```
Array.prototype.customMethod = function () {
  let sum = 0;
  for (let i = 0; i < this.length; i++) {
    sum += this[i];
  }
  return sum;
};

const numbers = [1, 2, 3, 4, 5];
console.log(numbers.customMethod());
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr6.js"
15
```

Practical-7

Aim:-

Create a JavaScript module that exports a class representing a calculator with methods to perform basic arithmetic operations. Import the module in another JavaScript file and use the calculator class to perform calculations.

Code:-

```
const Calculator = require("./Pr7Export");

const calculator = new Calculator();

const resultAdd = calculator.add(10, 5);
console.log("10 + 5 =", resultAdd);

const resultSubtract = calculator.subtract(10, 5);
console.log("10 - 5 =", resultSubtract);

const resultMultiply = calculator.multiply(10, 5);
console.log("10 * 5 =", resultMultiply);

const resultDivide = calculator.divide(10, 5);
console.log("10 / 5 =", resultDivide);
```

Class File:-

```
class Calculator {
  add(a, b) {
    return a + b;
  }

  subtract(a, b) {
    return a - b;
  }

  multiply(a, b) {
    return a * b;
  }

  divide(a, b) {
    if (b === 0) {
      throw new Error("Cannot divide by zero");
    }
    return a / b;
```

```
  }
}

module.exports = Calculator;
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr7\Pr7.js"
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
PS E:\Darsh\AWT>
```

Practical-8

Aim:-

Create a JavaScript module that fetches data from an API using the fetch() function and exports the retrieved data.

Create an async function getUsers(names), that gets an array of GitHub logins, fetches the users from GitHub and returns an array of GitHub users.

The GitHub url with user information for the given USERNAME is: https://api.github.com/users/USERNAME.

There's a test example in the sandbox.

Important details:
- There should be one fetch request per user.
- Requests shouldn't wait for each other. So that the data arrives as soon as possible.
- If any request fails, or if there's no such user, the function should return null in the resulting array.

Code:-

```
const { getUsers } = require('./Pr8API.js');

async function fetchUserName() {
  const usernames = ["006Darsh", "Rishi2103", "VasuBhut"];

  try {
    const usersData = await getUsers(usernames);

    if (usersData.length > 0) {
      console.log("GitHub Users Data:", usersData);
    } else {
      console.log("No valid users found.");
    }
  } catch (error) {
    console.error("Error fetching GitHub users:", error);
  }
}

fetchUserName();
```

Api Fetch:-

```
const fetch = require("node-fetch");

async function fetchUserData(username) {
  const response = await fetch(`https://api.github.com/users/${username}`);

  if (!response.ok) {
    return null;
  }

  const userData = await response.json();
  return userData;
}

async function getUsers(names) {
  const promises = names.map((name) => fetchUserData(name));
  const userDataArray = await Promise.all(promises);
  return userDataArray.filter((data) => data !== null);
}

module.exports = { getUsers };
```

Output:-

```
PS C:\Users\Dell> node "e:\Darsh\AWT\Practical File\Pr4.js"
Factorial of 5 is: 120
Hello, Darsh! Welcome to our website.
PS C:\Users\Dell>
```

Practical-9

Aim:-

Implement dynamic imports using the import() function to load modules asynchronously based on certain conditions.

Code:-

```
const condition = true;

if (condition) {
  import("./module1.mjs")
    .then((module1) => {
      module1.Hello1();
    })
    .catch((error) => console.error("Error importing Module A:", error));
} else {
  import("./module2.mjs")
    .then((module2) => {
      module2.Hello2();
    })
    .catch((error) => console.error("Error importing Module B:", error));
}
```

Module 1:-

```
export function Hello1() {
  console.log("Hello from Module 1!");
}
```

Module 2:-

```
export function Hello2() {
  console.log("Hello from Module 2!");
}
```

Output:-

```
PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr9\Pr9.js"
Hello from Module 1!
```

Practical-10

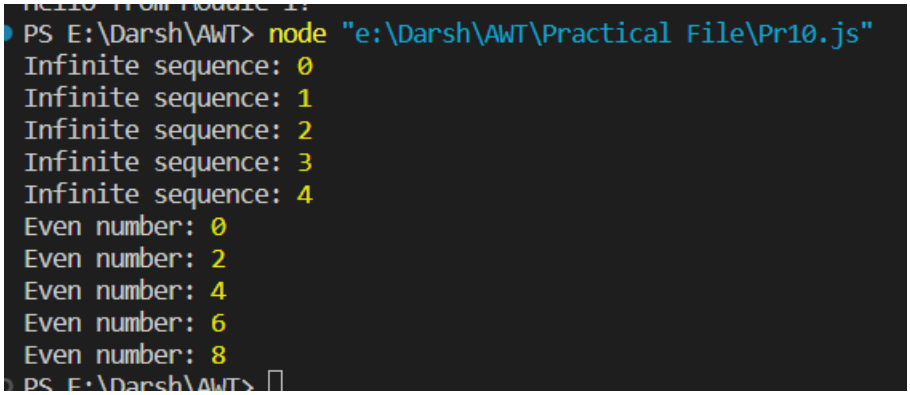
Aim:-

Create an iterator that generates an infinite sequence of numbers and a generator that yields a sequence of even numbers. Use the iterator and generator in different scenarios.

Code:-

```
function* infinite() {
  let i = 0;
  while (true) {
    yield i++;
  }
}

function* evenNumbers(count) {
  let i = 0;
  let iteratedCount = 0;

  while (iteratedCount < count) {
    if (i % 2 === 0) {
      yield i;
      iteratedCount++;
    }
    i++;
  }
}

const Iterator = infinite();
for (let i = 0; i < 5; i++) {
  console.log("Infinite sequence:", Iterator.next().value);
}

const evenNumbersGenerator = evenNumbers(5);
for (const evenNumber of evenNumbersGenerator) {
  console.log("Even number:", evenNumber);
}
```

Output:-


PS E:\Darsh\AWT> node "e:\Darsh\AWT\Practical File\Pr10.js"
Infinite sequence: 0
Infinite sequence: 1
Infinite sequence: 2
Infinite sequence: 3
Infinite sequence: 4
Even number: 0
Even number: 2
Even number: 4
Even number: 6
Even number: 8