**Question 1: What are lifecycle methods in React class components? Describe the phases of a component’s lifecycle.**

In React class components, lifecycle methods are special methods that are invoked at different stages of a component’s life in the DOM. They allow developers to run code at specific points in a component's creation, updating, and removal process.

**✅ The Component Lifecycle is Divided into 3 Main Phases:**

**1. Mounting Phase**

This is when the component is being created and inserted into the DOM.

**Main Lifecycle Methods:**

* constructor()  
  Initializes state and binds event handlers.
* static getDerivedStateFromProps(props, state)  
  Used to update state from props before rendering (rarely used).
* render()  
  Returns JSX to render UI.
* componentDidMount()  
  Called once after the component is mounted. Perfect for API calls, subscriptions, or DOM operations.

**2. Updating Phase**

Triggered when a component’s **props or state** changes.

**Main Lifecycle Methods:**

* static getDerivedStateFromProps(props, state)  
  Also called during updates.
* shouldComponentUpdate(nextProps, nextState)  
  Decides whether the component should re-render (returns true or false).
* render()  
  Called again to update the UI.
* getSnapshotBeforeUpdate(prevProps, prevState)  
  Captures some information (e.g., scroll position) before the DOM is updated.
* componentDidUpdate(prevProps, prevState, snapshot)  
  Called after the component updates. Good for handling side effects.

**3. Unmounting Phase**

When the component is **removed** from the DOM.

**Main Lifecycle Method:**

* componentWillUnmount()  
  Cleanup activities go here—like removing event listeners or cancelling API calls.

**Question 2: Explain the purpose of componentDidMount(), componentDidUpdate(), and componentWillUnmount().**

**🔹 componentDidMount()**

**When it runs:**  
Right after the component is **rendered to the DOM for the first time** (mounting phase).

**Purpose:**

* To perform actions **after** the initial render.
* Ideal for:
  + **API calls** to fetch data.
  + **Setting up subscriptions** or timers.
  + **Interacting with the DOM** (e.g., focus on an input).

**Example:**

componentDidMount() {

fetch("https://api.example.com/data")

.then(res => res.json())

.then(data => this.setState({ data }));

}

**🔹 componentDidUpdate(prevProps, prevState)**

**When it runs:**  
After the component **re-renders due to changes** in props or state.

**Purpose:**

* To respond to **updates** (like new props or state).
* Useful for:
  + **Reacting to prop changes**
  + **Making API calls** based on state changes
  + **Updating the DOM** based on new data

**Example:**

componentDidUpdate(prevProps, prevState) {

if (prevProps.userId !== this.props.userId) {

this.fetchUserData(this.props.userId);

}

}

**🔹 componentWillUnmount()**

**When it runs:**  
Right before the component is **removed from the DOM** (unmounting phase).

**Purpose:**

* To **clean up** things that could cause memory leaks.
* Ideal for:
  + **Removing event listeners**
  + **Clearing timers or intervals**
  + **Unsubscribing** from data sources

**Example:**

componentWillUnmount() {

clearInterval(this.timer);

window.removeEventListener('resize', this.handleResize);

}