```
// 1) Modelo de dominio con polimorfismo y herencia de ingredientes

// Clase abstracta: no se puede instanciar directamente, solo sirve como base
abstract class Tarta {

  // Declaramos las propiedades de la clase
public sabor: string;
public imagen: string;

// Constructor: asignamos los valores
constructor(sabor: string, imagen: string) {
  this.sabor = sabor;
  this.imagen = imagen;
}

  // Método abstracto: cada clase hija debe implementar su propio hornear()
  abstract hornear(): string;

  // Método abstracto protegido: solo accesible desde la clase y sus hijas
  protected abstract ingredientesEspecificos(): string[];

  // Método concreto que devuelve todos los ingredientes
  // Combina ingredientes comunes + los específicos de cada tarta
  ingredientes(): string[] {
    const base = ["Harina", "Huevos", "Azúcar"]; // ingredientes comunes
    return [...base, ...this.ingredientesEspecificos()]; // spread operator combina arrays
  }

  // Getter: permite acceder a this.nombre como si fuera propiedad
  // 'this' hace referencia a la instancia concreta de la clase
  get nombre(): string {
    return `Tarta de ${this.sabor}`;
  }
}

// -------------------- CLASES CONCRETAS --------------------
// 'extends Tarta' indica que estas clases heredan de Tarta
// Heredan propiedades y métodos, y deben implementar los abstractos
```

```
class TartaChocolate extends Tarta {
  // constructor llama al de la clase padre usando 'super()'
  constructor() { super("chocolate", "/chocolate.png"); }

  // Implementación del método abstracto hornear()
  hornear(): string {
    // usamos this.nombre para acceder al getter de esta instancia
    return `🍫 ${this.nombre}: mezclando cacao… Horneando a 180°C
durante 30 min. ¡Lista!`;
  }
  // this.nombre es un getter, se calcula al acceder, no se almacena,
por lo que no necesitamos declararla

  // Implementación de los ingredientes específicos
  protected ingredientesEspecificos(): string[] {
    return ["Cacao", "Mantequilla"];
  }
}

class TartaFresa extends Tarta {
  constructor() { super("fresa", "/fresa.png"); }
  hornear(): string {
    return `🍓 ${this.nombre}: base esponjosa… Horneando a 170°C
durante 25 min. ¡Lista!`;
  }
  protected ingredientesEspecificos(): string[] {
    return ["Fresas frescas", "Nata"];
  }
}

class TartaQueso extends Tarta {
  constructor() { super("queso", "/queso.png"); }
  hornear(): string {
    return `🧀 ${this.nombre}: baño María… Horneando a 160°C durante 40
min. ¡Lista!`;
  }
  protected ingredientesEspecificos(): string[] {
    return ["Queso crema", "Galletas trituradas"];
  }
}

class TartaLimon extends Tarta {
```

```
  constructor() { super("limón", "/limon.png"); }
  hornear(): string {
    return `🍋 ${this.nombre}: ralladura de limón… Horneando a 175°C
durante 28 min. ¡Lista!`;
  }
  protected ingredientesEspecificos(): string[] {
    return ["Limón", "Ralladura de limón"];
  }
}

class TartaZanahoria extends Tarta {
  constructor() { super("zanahoria", "/zanahoria.png"); }
  hornear(): string {
    return `🥕 ${this.nombre}: rallando zanahorias… Horneando a 175°C
durante 35 min. ¡Lista!`;
  }
  protected ingredientesEspecificos(): string[] {
    return ["Zanahorias", "Canela", "Aceite"];
  }
}

// -------------------- ESTADO INICIAL --------------------

// Lista de tartas: comienza vacía para que la web se muestre limpia
const tartas: Tarta[] = [];

// En el Estado inicial: creamos un array con algunas tartas para
mostrar al cargar la página
// Cada elemento del array es una instancia de una clase concreta de
tarta
// Así, al hacer el primer render, la lista no está vacía y podemos ver
ejemplos

// *podríamos incluir new TartaChocolate() dentro de los corchetes del
array para que se muestre desde el render inicial

// -------------------- REFERENCIAS AL DOM --------------------
const lista = document.getElementById('lista') as HTMLDivElement; //
contenedor para mostrar las tartas
const resultados = document.getElementById('resultados') as
HTMLUListElement; // lista de resultados de horneado

// -------------------- FUNCIONES --------------------
```

```
// Renderiza la lista de tartas en la web
function renderLista(): void {
  // Reemplazamos el contenido del contenedor por el HTML generado
  lista.innerHTML = tartas
    .map(
      (t, i) => `
      <article class="card">
        <img src="${t.imagen}" alt="${t.nombre}" class="foto"/>
        <h3>${t.nombre}</h3>
        <p><small>Clase: <code>${t.constructor.name}</code></small></p>
        <p><strong>Ingredientes:</strong> ${t.ingredientes().join(",
")}</p>
        <button class="btn-primary" data-index="${i}">Hornear</button>
      </article>`
    )
    .join('');

  // Añadimos eventos a los botones "Hornear"
  document.querySelectorAll<HTMLButtonElement>('.card
.btn-primary').forEach((btn) => {
    btn.addEventListener('click', () => {
      const idx = Number(btn.dataset.index); // índice de la tarta
      const mensaje = tartas[idx].hornear(); // llamar al método
hornear() de esa instancia
      pushResultado(mensaje); // mostrar resultado
    });
  });
}

// Añade un resultado a la lista de horneado
function pushResultado(texto: string): void {
  const li = document.createElement('li'); // creamos un <li>
  li.textContent = texto; // asignamos el texto
  resultados.prepend(li); // añadimos al inicio de la lista
}

// -------------------- BOTONES --------------------

// Botón "Añadir tarta"
const btnAdd = document.getElementById('btn-add') as HTMLButtonElement;
btnAdd.addEventListener('click', () => {
  const select = document.getElementById('tipo') as HTMLSelectElement;
```

```
  const tipo = select.value; // obtenemos el tipo seleccionado
  const nueva = crearTarta(tipo); // creamos la tarta correspondiente
  tartas.push(nueva); // añadimos al array
  renderLista(); // renderizamos la lista de nuevo
});

// Botón "Hornear todas"
const btnHornearTodas = document.getElementById('btn-hornear-todas') as
HTMLButtonElement;
btnHornearTodas.addEventListener('click', () => {
  tartas.forEach((t) => pushResultado(t.hornear())); // llamamos
hornear() a cada tarta
});

// Función que devuelve una instancia de Tarta según el tipo
function crearTarta(tipo: string): Tarta {
  switch (tipo) {
    case 'chocolate': return new TartaChocolate();
    case 'fresa': return new TartaFresa();
    case 'queso': return new TartaQueso();
    case 'limon': return new TartaLimon();
    case 'zanahoria': return new TartaZanahoria();
    default: return new TartaChocolate(); // fallback
  }
}

// -------------------- PRIMER RENDER --------------------

// Mostrar lista inicial (vacía) al cargar la web
renderLista();
```