```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>Java Programming for Beginners (Offline)</title>
 <style>
  /* Minimal Tailwind-like styles (custom) */
  :root {
   --bg-gradient: linear-gradient(to bottom right, #eff6ff, #e0f2fe);
   --card-bg: #ffffff;
   --blue-600: #2563eb;
   --blue-700: #1d4ed8;
   --gray-100: #f3f4f6;
   --gray-200: #e5e7eb;
   --gray-400: #9ca3af;
   --gray-600: #4b5563;
   --gray-700: #374151;
   --gray-800: #1f2937;
   --gray-900: #111827;
   --yellow-50: #fefce8;
   --yellow-400: #facc15;
   --yellow-600: #ca8a04;
   --green-50: #f0fdf4;
   --green-600: #16a34a;
   --orange-50: #fff7ed;
   --orange-600: #ea580c;
   --purple-50: #f5f3ff;
   --purple-600: #7c3aed;
   --pink-50: #fdf2f8;
   --pink-600: #db2777;
   --indigo-50: #eef2ff;
   --indigo-600: #4f46e5;
   --teal-50: #f0fdfa;
   --teal-600: #0d9488;
   --amber-50: #fffbeb;
   --amber-100: #fef3c7;
  }

  * {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
  }
```

```
body {
 font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
 background: var(--bg-gradient);
 padding: 1.5rem;
 color: var(--gray-800);
}

.max-w-6xl {
 max-width: 72rem;
 margin: 0 auto;
}

.card {
 background: var(--card-bg);
 border-radius: 0.5rem;
 box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
 padding: 1.5rem;
 margin-bottom: 1.5rem;
}

.section-header {
 display: flex;
 align-items: center;
 gap: 0.75rem;
 margin-bottom: 1.5rem;
 padding-bottom: 1rem;
 border-bottom: 2px solid #dbeafe;
}

.icon {
 width: 32px;
 height: 32px;
 display: inline-flex;
 align-items: center;
 justify-content: center;
 font-weight: bold;
 font-size: 1.25rem;
}

.blue-icon { color: var(--blue-600); }
.yellow-icon { color: var(--yellow-600); }
.green-icon { color: var(--green-600); }

h1 {
 font-size: 1.875rem;
```

```
 font-weight: 700;
 color: var(--gray-800);
}

h2 {
 font-size: 1.5rem;
 font-weight: 700;
}

h3 {
 font-weight: 700;
 margin-bottom: 0.75rem;
}

p, li, span {
 line-height: 1.6;
}

.btn {
 padding: 0.5rem 1rem;
 border-radius: 0.5rem;
 font-size: 0.875rem;
 font-weight: 600;
 cursor: pointer;
 display: inline-flex;
 align-items: center;
 gap: 0.375rem;
 transition: all 0.2s;
}

.btn-nav {
 background: var(--gray-100);
 color: var(--gray-700);
}

.btn-nav:hover:not(:disabled) {
 background: #d1d5db;
}

.btn-nav.active {
 background: var(--blue-600);
 color: white;
 box-shadow: 0 4px 6px -1px rgba(37, 99, 235, 0.3);
}
```

```
.btn-primary {
 background: var(--blue-600);
 color: white;
}

.btn-primary:hover:not(:disabled) {
 background: var(--blue-700);
}

.btn-disabled {
 background: var(--gray-200);
 color: var(--gray-400);
 cursor: not-allowed;
}

.analogy-box {
 background: var(--yellow-50);
 border-left: 4px solid var(--yellow-400);
 padding: 1.5rem;
 border-radius: 0 0.5rem 0.5rem 0;
 margin-bottom: 1.5rem;
}

.key-points {
 display: grid;
 grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
 gap: 0.75rem;
 margin-bottom: 1.5rem;
}

.key-point {
 display: flex;
 gap: 0.5rem;
 background: var(--green-50);
 padding: 0.75rem;
 border-radius: 0.5rem;
 font-size: 0.875rem;
 color: var(--gray-700);
}

.code-block {
 background: var(--gray-900);
 color: #f9fafb;
 padding: 1.5rem;
 border-radius: 0.5rem;
```

```
 overflow-x: auto;
 font-family: ui-monospace, SFMono-Regular, 'Monaco', monospace;
 font-size: 0.875rem;
 white-space: pre;
 margin-bottom: 1.5rem;
}

.visual {
 margin-bottom: 1.5rem;
}

.nav-buttons {
 display: flex;
 justify-content: space-between;
 align-items: center;
 padding-top: 1.5rem;
 border-top: 2px solid var(--gray-200);
}

.section-footer {
 text-align: center;
}

/* Flex & Grid Helpers */
.flex { display: flex; }
.flex-col { flex-direction: column; }
.items-center { align-items: center; }
.justify-center { justify-content: center; }
.justify-between { justify-content: space-between; }
.gap-2 { gap: 0.5rem; }
.gap-3 { gap: 0.75rem; }
.gap-4 { gap: 1rem; }
.gap-6 { gap: 1.5rem; }
.gap-8 { gap: 2rem; }
.text-center { text-align: center; }
.mb-2 { margin-bottom: 0.5rem; }
.mb-3 { margin-bottom: 0.75rem; }
.mb-4 { margin-bottom: 1rem; }
.mb-6 { margin-bottom: 1.5rem; }
.mt-1 { margin-top: 0.25rem; }
.mt-2 { margin-top: 0.5rem; }
.w-16 { width: 4rem; }
.h-16 { height: 4rem; }
.w-20 { width: 5rem; }
.h-20 { height: 5rem; }
```

```
  .rounded { border-radius: 0.25rem; }
  .rounded-lg { border-radius: 0.5rem; }
  .rounded-full { border-radius: 9999px; }
  .font-semibold { font-weight: 600; }
  .font-bold { font-weight: 700; }
  .text-sm { font-size: 0.875rem; }
  .text-lg { font-size: 1.125rem; }
  .text-xl { font-size: 1.25rem; }
  .text-2xl { font-size: 1.5rem; }
  .text-3xl { font-size: 1.875rem; }
  .text-4xl { font-size: 2.25rem; }
  .text-6xl { font-size: 3.75rem; }

  /* Responsive */
  @media (max-width: 768px) {
   .key-points {
    grid-template-columns: 1fr;
   }
   body {
    padding: 1rem;
   }
   .card {
    padding: 1rem;
   }
  }
 </style>
</head>
<body>
 <div class="max-w-6xl">
  <!-- Header -->
  <div class="card">
   <div class="flex items-center gap-3 mb-2">
    <span class="icon" style="color: #ea580c;">☕</span>
    <h1>Java Programming for Beginners</h1>
   </div>
   <p>Learn Java from scratch with easy explanations, visuals, and hands-on examples!</p>
  </div>

  <!-- Navigation Buttons -->
  <div class="card" id="navButtons"></div>

  <!-- Content Area -->
  <div class="card" id="contentArea"></div>

  <!-- Footer -->
```

```
  <div class="card section-footer">
   <h3>🎯 Practice Makes Perfect!</h3>
   <p>The best way to learn Java is by writing code. Try modifying the examples above and see what happens!</p>
  </div>
 </div>

 <script>
  const sections = [
   {
    id: 0,
    title: "What is Java?",
    icon: "☕",
    content: {
     analogy: "Think of Java as a universal language that computers around the world understand. Just like English helps people from different countries communicate, Java helps programs run on any computer - whether it's Windows, Mac, or Linux!",
     visual: "coffee-intro",
     keyPoints: [
      "Java is a programming language created in 1995",
      "Write once, run anywhere (WORA)",
      "Used for websites, mobile apps, games, and more",
      "One of the most popular languages in the world"
     ],
     code: `// Your first Java program
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}`
    }
   },
   {
    id: 1,
    title: "Variables & Data Types",
    icon: "📦",
    content: {
     analogy: "Variables are like labeled boxes where you store different things. Just like you need different sized boxes for shoes, books, or toys, Java has different data types for numbers, text, and more!",
     visual: "variables",
     keyPoints: [
      "Variables store data in your program",
      "Each variable has a type (int, String, boolean, etc.)",
      "Must declare before using",
```

```
        "Java is 'strongly typed' - types matter!"
       ],
       code: `// Different types of variables
int age = 25;         // Whole numbers
double price = 19.99;    // Decimal numbers
String name = "John";    // Text
boolean isStudent = true;  // True or false
char grade = 'A';       // Single character
// Using variables
System.out.println("Name: " + name);
System.out.println("Age: " + age);
System.out.println("Price: $" + price);`
      }
     },
     {
      id: 2,
      title: "Operators",
      icon: "< >+-",
      content: {
       analogy: "Operators are like the mathematical symbols you learned in school (+, -, ×, ÷), but Java has even more! They help you perform operations on your data.",
       visual: "operators",
       keyPoints: [
        "Arithmetic: +, -, *, /, % (remainder)",
        "Comparison: ==, !=, <, >, <=, >=",
        "Logical: && (and), || (or), ! (not)",
        "Assignment: =, +=, -=, *=, /="
       ],
       code: `// Arithmetic operators
int a = 10;
int b = 3;
System.out.println("Sum: " + (a + b));      // 13
System.out.println("Difference: " + (a - b)); // 7
System.out.println("Product: " + (a * b));   // 30
System.out.println("Division: " + (a / b));  // 3
System.out.println("Remainder: " + (a % b));  // 1
// Comparison operators
System.out.println(a > b);  // true
System.out.println(a == b);  // false
// Logical operators
boolean hasLicense = true;
int age = 20;
boolean canDrive = hasLicense && (age >= 18);
System.out.println("Can drive: " + canDrive); // true`
      }
```

```
    },
    {
     id: 3,
     title: "Control Flow: If-Else",
     icon: "💡",
     content: {
      analogy: "If-else statements are like making decisions in real life. 'If it's raining, take an umbrella,
else wear sunglasses.' Your program makes choices based on conditions!",
      visual: "control-flow",
      keyPoints: [
       "If statements check conditions",
       "Code runs only if condition is true",
       "Else provides an alternative path",
       "Can chain with else-if for multiple conditions"
      ],
      code: `// Simple if-else
int temperature = 25;
if (temperature > 30) {
  System.out.println("It's hot outside!");
} else if (temperature > 20) {
  System.out.println("Nice weather!");
} else {
  System.out.println("It's cold!");
}
// Real-world example: Grade calculator
int score = 85;
char grade;
if (score >= 90) {
  grade = 'A';
} else if (score >= 80) {
  grade = 'B';
} else if (score >= 70) {
  grade = 'C';
} else {
  grade = 'F';
}
System.out.println("Your grade: " + grade);`
     }
    },
    {
     id: 4,
     title: "Loops",
     icon: "▶️",
     content: {
      analogy: "Loops are like doing repetitive tasks. Instead of writing 'wash dish' 10 times, you say 'wash
```

```
dishes while there are dishes left.' Loops help you repeat code efficiently!",
      visual: "loops",
      keyPoints: [
       "For loop: when you know how many times to repeat",
       "While loop: repeat while condition is true",
       "Do-while: execute at least once, then check condition",
       "Break and continue control loop execution"
      ],
      code: `// For loop - count from 1 to 5
for (int i = 1; i <= 5; i++) {
  System.out.println("Count: " + i);
}
// While loop - keep asking until correct
int password = 0;
int correctPassword = 1234;
while (password != correctPassword) {
  System.out.println("Enter password:");
  password = 1234; // Simulate user input
}
System.out.println("Access granted!");
// Enhanced for loop - iterate through array
String[] fruits = {"Apple", "Banana", "Orange"};
for (String fruit : fruits) {
  System.out.println("I like " + fruit);
}
// Break example
for (int i = 1; i <= 10; i++) {
  if (i == 5) break;
  System.out.println(i);
} // Stops at 4`
    }
   },
   {
    id: 5,
    title: "Arrays",
    icon: "📦",
    content: {
     analogy: "An array is like a row of lockers, all numbered. Each locker can hold one item. Instead of
having separate variables for student1, student2, student3, you have one 'students' array with
numbered positions!",
      visual: "arrays",
      keyPoints: [
       "Arrays store multiple values of the same type",
       "Fixed size once created",
       "Index starts at 0 (first element is at position 0)",
```

```
        "Access elements using square brackets []"
      ],
      code: `// Creating arrays
int[] numbers = {10, 20, 30, 40, 50};
String[] names = new String[3];
// Assigning values
names[0] = "Alice";
names[1] = "Bob";
names[2] = "Charlie";
// Accessing elements
System.out.println("First number: " + numbers[0]); // 10
System.out.println("First name: " + names[0]);    // Alice
// Array length
System.out.println("Array size: " + numbers.length); // 5
// Looping through array
System.out.println("All numbers:");
for (int i = 0; i < numbers.length; i++) {
  System.out.println(numbers[i]);
}
// Finding maximum in array
int max = numbers[0];
for (int num : numbers) {
  if (num > max) {
    max = num;
  }
}
System.out.println("Maximum: " + max);`
    }
   },
   {
    id: 6,
    title: "Methods (Functions)",
    icon: "{}",
    content: {
     analogy: "Methods are like recipes. You give them ingredients (parameters), they follow steps, and give you back a dish (return value). Instead of writing the same recipe every time, you just call it by name!",
     visual: "methods",
     keyPoints: [
      "Methods organize code into reusable blocks",
      "Can take inputs (parameters) and return outputs",
      "Must specify return type (or void for no return)",
      "Call methods by name with arguments"
     ],
     code: `// Method that returns a value
```

```
public static int addNumbers(int a, int b) {
  int sum = a + b;
  return sum;
}
// Method with no return value (void)
public static void greetPerson(String name) {
  System.out.println("Hello, " + name + "!");
}
// Method with multiple parameters
public static double calculateArea(double length, double width) {
  return length * width;
}
// Method overloading - same name, different parameters
public static int multiply(int a, int b) {
  return a * b;
}
public static double multiply(double a, double b) {
  return a * b;
}
// Using the methods
public static void main(String[] args) {
  int result = addNumbers(5, 3);
  System.out.println("Sum: " + result);
  greetPerson("Alice");
  double area = calculateArea(5.0, 3.0);
  System.out.println("Area: " + area);
  System.out.println(multiply(4, 5));    // 20
  System.out.println(multiply(4.5, 2.0)); // 9.0
} `
    }
   },
   {
    id: 7,
    title: "Classes & Objects (OOP)",
    icon: "👥",
    content: {
     analogy: "Think of a Class as a blueprint for a house, and Objects as the actual houses built from that blueprint. The blueprint defines what every house has (rooms, doors), but each house is a separate, real thing with its own specific details!",
     visual: "oop",
     keyPoints: [
      "Class is a template, Object is an instance",
      "Classes have properties (fields) and behaviors (methods)",
      "Objects are created using 'new' keyword",
      "Encapsulation: keep data safe with private/public"
```

```
      ],
      code: `// Defining a class
public class Car {
  // Properties (fields)
  private String brand;
  private String color;
  private int year;
  // Constructor - runs when object is created
  public Car(String brand, String color, int year) {
    this.brand = brand;
    this.color = color;
    this.year = year;
  }
  // Methods (behaviors)
  public void start() {
    System.out.println(brand + " is starting...");
  }
  public void displayInfo() {
    System.out.println(year + " " + color + " " + brand);
  }
  // Getters and setters
  public String getBrand() {
    return brand;
  }
  public void setColor(String color) {
    this.color = color;
  }
}
// Using the class
public class Main {
  public static void main(String[] args) {
    // Creating objects
    Car myCar = new Car("Toyota", "Red", 2022);
    Car yourCar = new Car("Honda", "Blue", 2023);
    // Using object methods
    myCar.start();
    myCar.displayInfo();
    yourCar.start();
    yourCar.displayInfo();
  }
}`
    }
   }
  ];
```

```
  let activeSection = 0;

  function renderVisual(type) {
   const visuals = {
    'coffee-intro': `
     <div class="flex items-center justify-center gap-8 p-8" style="background: linear-gradient(to
right, var(--amber-50), var(--amber-100)); border-radius: 0.5rem;">
      <div class="text-center">
       <div class="text-6xl mb-2">☕</div>
       <div class="font-semibold">Java Code</div>
      </div>
      <div class="text-4xl">→</div>
      <div class="text-center">
       <div class="text-6xl mb-2">💻</div>
       <div class="font-semibold">Any Computer</div>
      </div>
      <div class="text-4xl">→</div>
      <div class="text-center">
       <div class="text-6xl mb-2">✨</div>
       <div class="font-semibold">Working Program</div>
      </div>
     </div>
    `,
    variables: `
     <div class="grid grid-cols-2 gap-4 p-6" style="background: var(--blue-50); border-radius:
0.5rem;">
      <div class="bg-white p-4 rounded shadow text-center">
       <div class="text-3xl mb-2">📦</div>
       <div class="font-bold" style="color: var(--blue-600);">int age = 25</div>
       <div class="text-sm" style="color: var(--gray-600); margin-top: 0.5rem;">Box labeled "age"
holds number 25</div>
      </div>
      <div class="bg-white p-4 rounded shadow text-center">
       <div class="text-3xl mb-2">📝</div>
       <div class="font-bold" style="color: #16a34a;">String name = "John"</div>
       <div class="text-sm" style="color: var(--gray-600); margin-top: 0.5rem;">Box labeled "name"
holds text "John"</div>
      </div>
      <div class="bg-white p-4 rounded shadow text-center">
       <div class="text-3xl mb-2">💰</div>
       <div class="font-bold" style="color: var(--purple-600);">double price = 19.99</div>
       <div class="text-sm" style="color: var(--gray-600); margin-top: 0.5rem;">Box for decimal
numbers</div>
      </div>
      <div class="bg-white p-4 rounded shadow text-center">
```

```
      <div class="text-3xl mb-2">✅</div>
      <div class="font-bold" style="color: #dc2626;">boolean isTrue = true</div>
      <div class="text-sm" style="color: var(--gray-600); margin-top: 0.5rem;">Box holds true or false</div>
     </div>
    </div>
  `,
  operators: `
    <div class="p-6" style="background: var(--purple-50); border-radius: 0.5rem;">
     <div class="grid grid-cols-3 gap-4 mb-4">
      <div class="bg-white p-3 rounded shadow text-center">
       <div class="text-2xl font-bold" style="color: var(--blue-600);">+</div>
       <div class="text-sm">Addition</div>
       <div class="text-xs" style="color: var(--gray-600);">5 + 3 = 8</div>
      </div>
      <div class="bg-white p-3 rounded shadow text-center">
       <div class="text-2xl font-bold" style="color: #16a34a;">-</div>
       <div class="text-sm">Subtraction</div>
       <div class="text-xs" style="color: var(--gray-600);">5 - 3 = 2</div>
      </div>
      <div class="bg-white p-3 rounded shadow text-center">
       <div class="text-2xl font-bold" style="color: var(--purple-600);">*</div>
       <div class="text-sm">Multiplication</div>
       <div class="text-xs" style="color: var(--gray-600);">5 * 3 = 15</div>
      </div>
     </div>
     <div class="grid grid-cols-3 gap-4">
      <div class="bg-white p-3 rounded shadow text-center">
       <div class="text-2xl font-bold" style="color: #dc2626;">==</div>
       <div class="text-sm">Equal to</div>
       <div class="text-xs" style="color: var(--gray-600);">5 == 5 → true</div>
      </div>
      <div class="bg-white p-3 rounded shadow text-center">
       <div class="text-2xl font-bold" style="color: #f97316;">&gt;</div>
       <div class="text-sm">Greater than</div>
       <div class="text-xs" style="color: var(--gray-600);">5 &gt; 3 → true</div>
      </div>
      <div class="bg-white p-3 rounded shadow text-center">
       <div class="text-2xl font-bold" style="color: #db2777;">&&</div>
       <div class="text-sm">AND</div>
       <div class="text-xs" style="color: var(--gray-600);">Both must be true</div>
      </div>
     </div>
    </div>
  `,
```

```
  'control-flow': `
    <div class="p-6" style="background: var(--green-50); border-radius: 0.5rem;">
      <div class="flex flex-col items-center gap-4">
        <div class="bg-yellow-200 p-4 rounded-lg font-semibold text-center w-48">
          Is temperature &gt; 30?
        </div>
        <div class="flex gap-8">
          <div class="flex flex-col items-center">
            <div class="text-xl font-bold" style="color: var(--green-600);">✓ YES</div>
            <div class="mt-2 bg-green-200 p-3 rounded">Print "It's hot!"</div>
          </div>
          <div class="flex flex-col items-center">
            <div class="text-xl font-bold" style="color: #dc2626;">✗ NO</div>
            <div class="mt-2 bg-blue-200 p-3 rounded">Check next condition</div>
          </div>
        </div>
      </div>
    </div>
  `,
  loops: `
    <div class="p-6" style="background: var(--orange-50); border-radius: 0.5rem;">
      <div class="text-center mb-4 font-semibold text-lg">Loop cycles through tasks</div>
      <div class="flex justify-center gap-4">
        <div class="bg-orange-200 p-4 rounded-full w-16 h-16 flex items-center justify-center font-bold
text-xl">1</div>
        <div class="bg-orange-200 p-4 rounded-full w-16 h-16 flex items-center justify-center font-bold
text-xl">2</div>
        <div class="bg-orange-200 p-4 rounded-full w-16 h-16 flex items-center justify-center font-bold
text-xl">3</div>
        <div class="bg-orange-200 p-4 rounded-full w-16 h-16 flex items-center justify-center font-bold
text-xl">4</div>
        <div class="bg-orange-200 p-4 rounded-full w-16 h-16 flex items-center justify-center font-bold
text-xl">5</div>
      </div>
      <div class="text-center mt-4 text-sm" style="color: var(--gray-600);">
        Instead of writing 5 separate commands, the loop does it automatically!
      </div>
    </div>
  `,
  arrays: `
    <div class="p-6" style="background: var(--pink-50); border-radius: 0.5rem;">
      <div class="text-center mb-4 font-semibold">Array: Like numbered lockers</div>
      <div class="flex justify-center gap-2">
        <div class="flex flex-col items-center">
          <div class="bg-pink-200 border-2 border-pink-400 p-3 rounded w-20 h-20 flex items-center
```

```
justify-center text-xs font-semibold">Apple</div>
        <div class="mt-1 text-sm font-bold" style="color: var(--pink-600);">[0]</div>
       </div>
       <div class="flex flex-col items-center">
        <div class="bg-pink-200 border-2 border-pink-400 p-3 rounded w-20 h-20 flex items-center
justify-center text-xs font-semibold">Banana</div>
        <div class="mt-1 text-sm font-bold" style="color: var(--pink-600);">[1]</div>
       </div>
       <div class="flex flex-col items-center">
        <div class="bg-pink-200 border-2 border-pink-400 p-3 rounded w-20 h-20 flex items-center
justify-center text-xs font-semibold">Orange</div>
        <div class="mt-1 text-sm font-bold" style="color: var(--pink-600);">[2]</div>
       </div>
       <div class="flex flex-col items-center">
        <div class="bg-pink-200 border-2 border-pink-400 p-3 rounded w-20 h-20 flex items-center
justify-center text-xs font-semibold">Grape</div>
        <div class="mt-1 text-sm font-bold" style="color: var(--pink-600);">[3]</div>
       </div>
       <div class="flex flex-col items-center">
        <div class="bg-pink-200 border-2 border-pink-400 p-3 rounded w-20 h-20 flex items-center
justify-center text-xs font-semibold">Mango</div>
        <div class="mt-1 text-sm font-bold" style="color: var(--pink-600);">[4]</div>
       </div>
      </div>
      <div class="text-center mt-4 text-sm" style="color: var(--gray-600);">
       Access any fruit by its index number: fruits[0] = "Apple"
      </div>
     </div>
    `,
    methods: `
     <div class="p-6" style="background: var(--indigo-50); border-radius: 0.5rem;">
      <div class="flex items-center justify-center gap-6">
       <div class="text-center">
        <div class="bg-indigo-200 p-4 rounded-lg">
         <div class="font-bold">Input</div>
         <div class="text-2xl">🥚 🥛 🍫</div>
         <div class="text-sm">ingredients</div>
        </div>
       </div>
       <div class="text-3xl">→</div>
       <div class="text-center">
        <div class="bg-indigo-300 p-4 rounded-lg">
         <div class="font-bold">Method</div>
         <div class="text-2xl">👨‍🍳</div>
         <div class="text-sm">makeCake()</div>
```

```
      </div>
     </div>
     <div class="text-3xl">→</div>
     <div class="text-center">
      <div class="bg-indigo-200 p-4 rounded-lg">
       <div class="font-bold">Output</div>
       <div class="text-2xl">🎂</div>
       <div class="text-sm">return cake</div>
      </div>
     </div>
    </div>
   </div>
  `,
  oop: `
   <div class="p-6" style="background: var(--teal-50); border-radius: 0.5rem;">
    <div class="grid grid-cols-2 gap-6">
     <div class="text-center">
      <div class="bg-teal-200 p-6 rounded-lg">
       <div class="text-3xl mb-2">📋</div>
       <div class="font-bold text-lg">Class (Blueprint)</div>
       <div class="text-sm mt-2">Car template:</div>
       <div class="text-xs mt-1">- brand</div>
       <div class="text-xs">- color</div>
       <div class="text-xs">- start()</div>
      </div>
     </div>
     <div class="flex flex-col gap-3">
      <div class="bg-teal-300 p-4 rounded-lg text-center">
       <div class="text-2xl mb-1">🚗</div>
       <div class="font-semibold text-sm">Object 1: Toyota</div>
      </div>
      <div class="bg-teal-300 p-4 rounded-lg text-center">
       <div class="text-2xl mb-1">🚙</div>
       <div class="font-semibold text-sm">Object 2: Honda</div>
      </div>
     </div>
    </div>
   </div>
  `
 };
 return visuals[type] || `<div>No visual available</div>`;
}

function renderNavButtons() {
 const container = document.getElementById('navButtons');
```

```
 container.innerHTML = `
  <div class="flex flex-wrap gap-2">
   ${sections.map(s => `
    <button data-id="${s.id}" class="btn btn-nav ${activeSection === s.id ? 'active' : ''}">
     <span>${s.icon}</span>
     <span class="text-sm font-medium">${s.title}</span>
    </button>
   `).join('')}
  </div>
 `;
 container.querySelectorAll('button').forEach(btn => {
  btn.addEventListener('click', () => {
   activeSection = parseInt(btn.dataset.id);
   renderNavButtons();
   renderContent();
  });
 });
}

function renderContent() {
 const sec = sections[activeSection];
 const content = `
  <div class="section-header">
   <span class="icon blue-icon">${sec.icon}</span>
   <h2>${sec.title}</h2>
  </div>

  <div class="analogy-box">
   <div class="flex items-start gap-3">
    <span class="icon yellow-icon">💡</span>
    <div>
     <h3>Simple Analogy</h3>
     <p>${sec.content.analogy}</p>
    </div>
   </div>
  </div>

  <div class="visual">
   <h3>Visual Representation</h3>
   ${renderVisual(sec.content.visual)}
  </div>

  <div class="key-points">
   <h3>Key Points to Remember</h3>
   ${sec.content.keyPoints.map(p => `
```

```
        <div class="key-point">
          <span class="icon green-icon">✓</span>
          <span>${p}</span>
        </div>
      `).join('')}
    </div>

    <div>
      <h3>Code Example</h3>
      <pre class="code-block">${sec.content.code}</pre>
    </div>

    <div class="nav-buttons">
      <button id="prevBtn" class="${activeSection === 0 ? 'btn btn-disabled' : 'btn btn-primary'}">
        ← Previous
      </button>
      <span class="text-sm" style="color: var(--gray-600);">
        Section ${activeSection + 1} of ${sections.length}
      </span>
      <button id="nextBtn" class="${activeSection === sections.length - 1 ? 'btn btn-disabled' : 'btn
btn-primary'}">
        Next →
      </button>
    </div>
  `;

  document.getElementById('contentArea').innerHTML = content;

  // Prev/Next logic
  document.getElementById('prevBtn')?.addEventListener('click', () => {
    if (activeSection > 0) {
      activeSection--;
      renderNavButtons();
      renderContent();
    }
  });
  document.getElementById('nextBtn')?.addEventListener('click', () => {
    if (activeSection < sections.length - 1) {
      activeSection++;
      renderNavButtons();
      renderContent();
    }
  });
}
```

```
  // Initialize
  renderNavButtons();
  renderContent();
 </script>
</body>
</html>
```