Class: ArrayStack

This class implements a basic stack data structure using an array.

Fields

- **int[] stack**: This array stores the elements of the stack.
- **int top**: This is an index that points to the top of the stack. It starts at -1, indicating the stack is empty.
- **int capacity:** This defines the maximum number of elements the stack can hold.

Constructor

```
public ArrayStack(int size) {
   stack = new int[size];
   capacity = size;
   top = -1;
}
```

The constructor initializes the stack with a given size. It sets the top index to -1, indicating that the stack is initially empty.

Methods

1. **Push method**

   ```
   public void push(int element) {
      if (isFull()) {
         System.out.println("Stack Overflow");
         return;
      }
      stack[++top] = element;
   }
   ```

   This method adds an element to the top of the stack.

   - First, it checks if the stack is full using the isFull() method.
   - If the stack is full, it prints "Stack Overflow" and returns.
   - If there is space, it increments the top index and assigns the element to the stack array at that position.

2. **Pop method**

```
public int pop() {
    if (isEmpty()) {
        System.out.println("Stack Underflow");
        return -1;
    }
    return stack[top--];
}
```

This method removes and returns the top element of the stack.

- It first checks if the stack is empty using the isEmpty() method.
- If the stack is empty, it prints "Stack Underflow" and returns -1.
- If the stack is not empty, it returns the element at the `top

3. **Peek method**

```
public int peek() {
    if (isEmpty()) {
        System.out.println("Stack is empty");
        return -1;
    }
    return stack[top];
}
```

This method returns the top element of the stack without removing it.

- It first checks if the stack is empty using the isEmpty() method.
- If the stack is empty, it prints "Stack is empty" and returns -1.
- If the stack is not empty, it returns the element at the top index.

4. **isEmpty method**

```
public boolean isEmpty() {
    return top == -1;
}
```

This method checks if the stack is empty.

- It returns true if the top index is -1 (indicating the stack is empty).
- Otherwise, it returns false.

5. **isFull method**

```
public boolean isFull() {
    return top == capacity - 1;
}
```

This method checks if the stack is full.

- It returns true if the top index is equal to capacity - 1 (indicating the stack is full).
- Otherwise, it returns false.

6. **Size method**

```
public int size() {
    return top + 1;
}
```

This method returns the number of elements in the stack.

- It returns top + 1 because the top index is zero-based (i.e., if top is 2, there are 3 elements in the stack).

7. **PrintStack method**

```
public void printStack() {
    if (isEmpty()) {
        System.out.println("Stack is empty");
        return;
    }
    for (int i = 0; i <= top; i++) {
        System.out.print(stack[i] + " ");
    }
    System.out.println();
}
```

This method prints all the elements in the stack from bottom to top.

- First, it checks if the stack is empty using the isEmpty() method.
- If the stack is empty, it prints "Stack is empty".
- If the stack is not empty, it iterates from the bottom of the stack (0) to the top index and prints each element.

main Method

```
public static void main(String[] args) {
    ArrayStack stack = new ArrayStack(5);

    stack.push(10);
    stack.push(20);
    stack.push(30);
    stack.push(40);
    stack.push(50);

    // Attempt to push when stack is full
    stack.push(60); // This will show "Stack Overflow"

    // Print stack elements
    System.out.print("Stack: ");
    stack.printStack();

    // Pop elements from stack
    System.out.println("Popped Element: " + stack.pop());

    // Peek at the top element
    System.out.println("Top Element: " + stack.peek());

    // Check if stack is empty
    System.out.println("Is Stack Empty? " + stack.isEmpty());

    // Check if stack is full
    System.out.println("Is Stack Full? " + stack.isFull());

    // Get the size of the stack
    System.out.println("Stack Size: " + stack.size());
}
```

This method demonstrates how to use the ArrayStack class:

- It creates a stack with a capacity of 5.
- It pushes five elements onto the stack.
- It attempts to push a sixth element, which fails and prints "Stack Overflow".
- It prints all elements in the stack.
- It pops the top element and prints it.
- It peeks at the top element and prints it.
- It checks and prints whether the stack is empty.
- It checks and prints whether the stack is full.

- It prints the current size of the stack.