Pseudo Random Number Generator

Aim:

To generate cryptographically secure random numbers, byte arrays, and Base64-encoded strings using SecureRandom in Java.

Description:

The program utilizes Java's SecureRandom class to generate secure random values. It provides methods for generating:

- Random integers
- Random integers within a specified range
- Random byte arrays
- Random Base64-encoded strings

This ensures strong randomness suitable for security-sensitive applications such as cryptographic key generation and token creation.

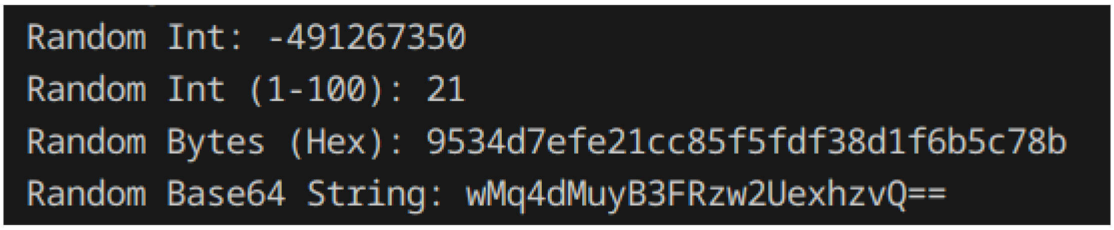
Code:

```
import java.security.NoSuchAlgorithmException;

import java.security.SecureRandom;

import java.util.Base64;


public class SecurePRNG {

    private SecureRandom secureRandom;


    public SecurePRNG() {

        try {

            // Use a strong algorithm like SHA1PRNG or NativePRNG

            this.secureRandom = SecureRandom.getInstanceStrong();

        } catch (NoSuchAlgorithmException e) {

            throw new RuntimeException("SecureRandom instance could not be created", e);

        }

    }


    // Generate a random integer

    public int getRandomInt() {
```

```
    return secureRandom.nextInt();
}

// Generate a random  integer within a range
public int getRandomIntInRange(int min, int max) {
    return min + secureRandom.nextInt(max - min + 1);
}

// Generate a random byte array
public byte[] getRandomBytes(int length) {
    byte[] bytes = new byte[length];
    secureRandom.nextBytes(bytes);
    return bytes;
}

// Generate a random string (Base64 encoded)
public String getRandomBase64String(int byteLength) {
    return Base64.getEncoder().encodeToString(getRandomBytes(byteLength));
}

public static void main(String[] args) {
    SecurePRNG prng = new SecurePRNG();

    System.out.println("Random Int: " + prng.getRandomInt());
    System.out.println("Random Int (1-100): " + prng.getRandomIntInRange(1, 100));
    System.out.println("Random Bytes (Hex): " + bytesToHex(prng.getRandomBytes(16)));
    System.out.println("Random Base64 String: " + prng.getRandomBase64String(16));
}

// Utility function to convert bytes to hex
private static String bytesToHex(byte[] bytes) {
    StringBuilder hexString = new StringBuilder();
```

```
        for (byte b : bytes) {
            hexString.append(String.format("%02x", b));
        }
        return hexString.toString();
    }
}
```

Output:



```
Random Int: -491267350
Random Int (1-100): 21
Random Bytes (Hex): 9534d7efe21cc85f5fdf38d1f6b5c78b
Random Base64 String: wMq4dMuyB3FRzw2UexhzvQ==
```

Code Explanation:

1. **Initialization (SecurePRNG constructor)**:
    - Creates a SecureRandom instance using SecureRandom.getInstanceStrong(), which selects a strong PRNG algorithm like SHA1PRNG or NativePRNG.
    - If an exception occurs due to the absence of a strong algorithm, it throws a runtime exception.
2. **Random Integer Generation (getRandomInt)**:
    - Uses secureRandom.nextInt() to generate a random integer.
3. **Random Integer in Range (getRandomIntInRange)**:
    - Computes a random integer within a specified range [min, max] using min + secureRandom.nextInt(max - min + 1).
4. **Random Byte Array (getRandomBytes)**:
    - Fills a byte array with random values using secureRandom.nextBytes(byteArray).
5. **Random Base64 String (getRandomBase64String)**:
    - Converts a random byte array into a Base64-encoded string using Base64.getEncoder().encodeToString().
6. **Main Method Execution**:
    - Calls and prints results from all the implemented methods.
    - Converts a random byte array to a hexadecimal string using bytesToHex() for readability.

Time Complexity:

- getRandomInt(): **O(1)**
- getRandomIntInRange(int min, int max): **O(1)**
- getRandomBytes(int length): **O(n)** (where n is the length of the byte array)
- getRandomBase64String(int byteLength): **O(n)** (since it internally calls getRandomBytes(n))
- bytesToHex(byte[]): **O(n)** (iterates through the byte array to convert it to hex)

Space Complexity:

- getRandomInt(): **O(1)**
- getRandomIntInRange(int min, int max): **O(1)**
- getRandomBytes(int length): **O(n)** (stores the byte array)
- getRandomBase64String(int byteLength): **O(n)** (stores both the byte array and Base64 string)
- bytesToHex(byte[]): **O(n)** (stores the hex string)