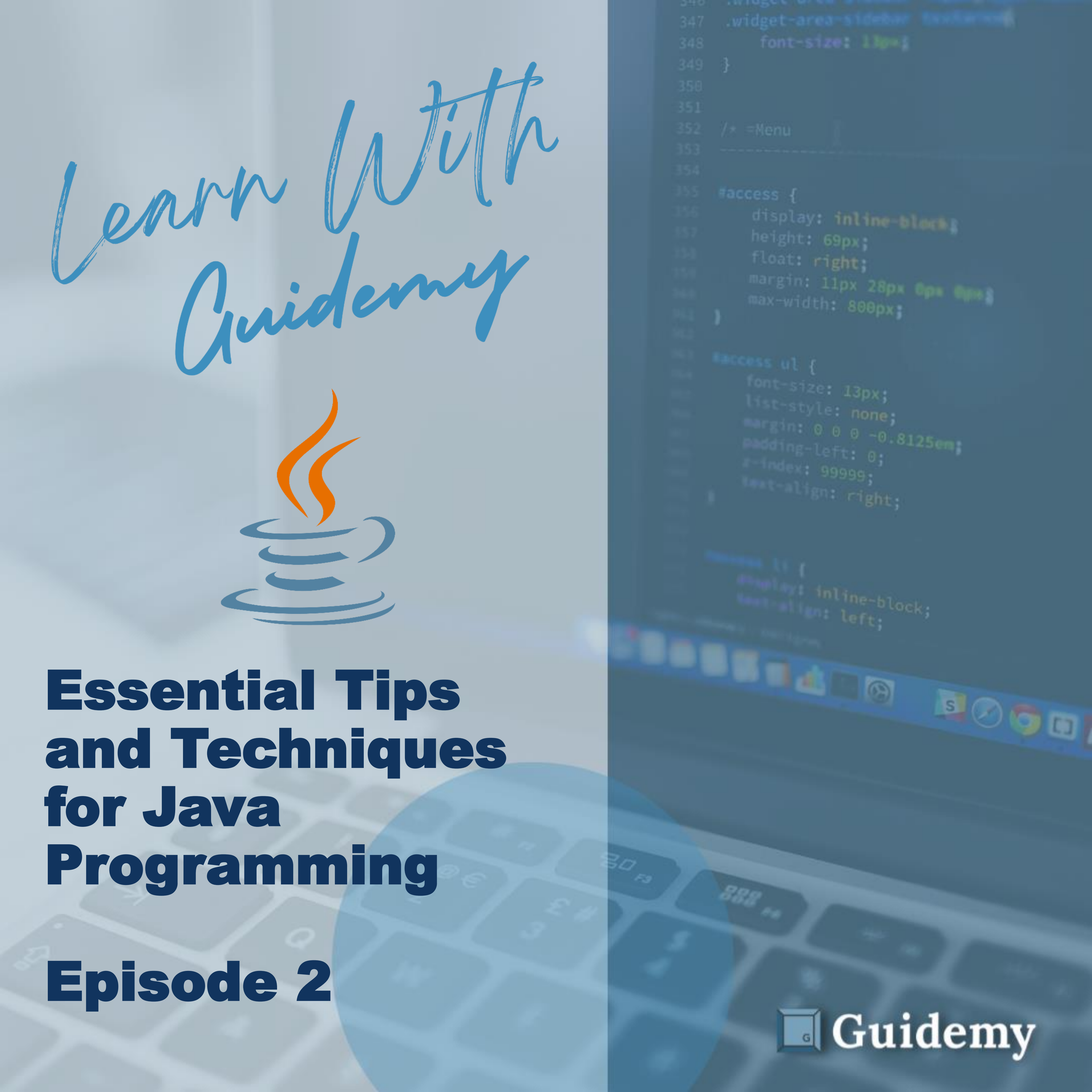

Essential Tips and Techniques for Java Programming

Episode 2

Guidemy

1. Use Lazy Loading

- Lazy loading strategy can be used where the object is created only when needed. For example:

```java
Person person = new Person();
if (isValid) {
  list.add(person);
}
```

- It is recommended to replace the code as follows:

```java
if (isValid) {
  Person person = new Person();
  list.add(person);
}
```

2. Avoid Overusing Static Variables

- It is important to note that when an object is referenced by a static variable, the garbage collector (GC) usually does not reclaim the heap memory occupied by that object. For example:

```java
public class A {
  private static B b = new B();
}
```

- In this case, the lifetime of the static variable `b` is the same as that of class `A`. If class `A` is not unloaded, the object referenced by `b` will remain in memory until the program terminates.

3. Specify the Initial Capacity

- By specifying the initial capacity of an array or a Collection, we can **avoid the overhead of resizing** the underlying data structure as new elements are added.
- This can improve the performance of our program, especially if we know **in advance** how many elements the array or the Collection is likely to hold.

```java
List<String> items = new ArrayList<>(1000);
```

4. Use Exceptions Cautiously

- Exceptions are **not performance-friendly**.
- When an exception is thrown, a new object is created. The `fillInStackTrace()` method of the `Throwable` interface, a `synchronized` method, is invoked. This method checks the stack and collects call trace information.
- Exceptions should only be used for error handling and **not for controlling the program flow**.

5. Avoid Using Synchronized Collections Unless Necessary

- When multiple threads access synchronized collections, they need to acquire locks on the collection to prevent race conditions and ensure consistency. This locking mechanism **slows down the performance** of the application as it involves context switching and overhead of acquiring and releasing locks.
- If thread safety is not a requirement, it is recommended to use `HashMap`, `ArrayList`, and `StringBuilder` instead of `ConcurrentHashMap`, `SynchronizedCollection`, and `StringBuffer`.

Summary

1. Use Lazy Loading strategy to create an object only when needed.
2. Avoid overusing static variables.
3. Specify the initial capacity of an array or a Collection to avoid the overhead of resizing.
4. Use Exceptions cautiously, as they're not performance-friendly.
5. Avoid using synchronized collections unless thread-safety is concerned.