RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

# Department of Master of Computer Applications (MCA)

# Mobile Application Development (MCA221IA)

# Hand Notes

| Unit - 1 | Topic: Storing Simple Data |
|---|---|
| **Editor:** 1RV24MC097 – Selvadiwakar G | |

## List of Questions

### 02 Marks Question

1) What is Storing Simple Data in Android?

### 04 Marks Question

2) What is SharedPreferences and how is it used?

### 06 Marks Question

3) Explain how data is written and read using Internal Storage in Android.

### 08 Marks Questions

4) Compare SharedPreferences and Internal Storage with examples.
5) Explain advantages and limitations of using SharedPreferences and Internal Storage.

### 10 Marks Question

6) Explain in detail the various methods of storing simple data in Android with examples.

## 02 Marks Question:

**1. What is Storing Simple Data in Android?**

**Answer:**

Storing Simple Data refers to saving small pieces of data like user preferences, flags, and configuration settings within an Android app. It helps maintain app state even after the app is closed or restarted.

**Example:**

Saving a user's theme preference (dark/light mode).

## 04 Marks Question:

**2. What is SharedPreferences and how is it used?**
**Answer:**
SharedPreferences is an Android API that stores key-value pairs of primitive data types (String, int, boolean, etc.) in a private XML file.

**Steps to use:**

1. Get SharedPreferences instance using `getSharedPreferences()` or `getPreferences()`.
2. Create an `Editor` object.
3. Put data using methods like `putString`, `putBoolean`.
4. Commit changes using `apply()` or `commit()`.

**Code Example:**

java
CopyEdit
```
SharedPreferences prefs = getSharedPreferences("MyPrefs", MODE_PRIVATE);
SharedPreferences.Editor editor = prefs.edit();
editor.putBoolean("isLoggedIn", true);
editor.apply();
```

**Use Case:**
Storing login status, app settings, or simple flags.

## 06 Marks Question:

**3. Explain how data is written and read using Internal Storage in Android.**
**Answer:**
Internal Storage allows an app to save files directly in the device's private storage, inaccessible to other apps. It's suitable for saving custom data like logs, small files, or user-generated content.

**Steps to Write Data:**

java
CopyEdit
```
FileOutputStream fos = openFileOutput("myfile.txt", MODE_PRIVATE);
fos.write("Hello Android!".getBytes());
fos.close();
```

**Steps to Read Data:**

java
CopyEdit

```
FileInputStream fis = openFileInput("myfile.txt");
BufferedReader br = new BufferedReader(new InputStreamReader(fis));
StringBuilder sb = new StringBuilder();
String line;
while ((line = br.readLine()) != null) {
    sb.append(line);
}
fis.close();
```

**Example:**
Saving form inputs or draft notes internally within the app.

## 08 Marks Question

**4. Compare SharedPreferences and Internal Storage with examples.**
**Answer:**

| Feature | SharedPreferences | Internal Storage |
|---|---|---|
| Data Type | Primitive types | Any type (e.g., text, binary) |
| Structure | Key-value pairs | Unstructured raw file data |
| Use Case | App settings, flags | Logs, documents, user files |
| Access Speed | Very fast (in-memory caching) | Fast, uses file I/O |
| Security | Private to app | Private to app |
| Auto Delete | Yes (on uninstall) | Yes (on uninstall) |

**Examples:**

- SharedPreferences: Remembering if user is logged in.
- Internal Storage: Saving app-generated logs or settings files.

**5. Explain advantages and limitations of using SharedPreferences and Internal Storage. Answer:**
**SharedPreferences:**

- Advantages: Lightweight, easy to use, fast.
- Limitations: Only primitive data types supported.

**Internal Storage:**

- Advantages: Secure, flexible (supports any file type).
- Limitations: Manual file handling, I/O complexity.

## 10 Marks Question

**6. Explain in detail the various methods of storing simple data in Android with examples.**
**Answer:**

**A. SharedPreferences**
Used to store key-value pairs for primitive data types.

**Code:**

java
CopyEdit

```
SharedPreferences sp = getSharedPreferences("UserPrefs", MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
editor.putString("username", "Dinesh");
editor.putBoolean("isDarkMode", true);
editor.apply();
```

**B. Internal Storage**
Used to save app-specific files privately.

**Code:**

java
CopyEdit

```
FileOutputStream fos = openFileOutput("settings.txt", MODE_PRIVATE);
fos.write("Dark mode ON".getBytes());
fos.close();
```

**C. External Storage (Optional Mention)**
Used for saving files accessible outside the app (e.g., images, PDFs). Requires permission.

**Use Cases:**

- SharedPreferences: Save simple settings.
- Internal Storage: Save temporary notes or JSON configs.

**Diagram (Optional):**

scss

CopyEdit

```
App → SharedPreferences (key-value)
App → Internal Storage (files)
```

**Example:**
A note-taking app:

- Saves theme and user login with SharedPreferences.
- Stores user notes in Internal Storage as text files.

}

```
// Inside sw.js
self.addEventListener('install', event => {
  console.log('Installing SW...');
  self.skipWaiting();
});

self.addEventListener('activate', event => {
  console.log('Activating SW...');
  event.waitUntil(clients.claim());
});

self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

(Lifecycle):

Client → Register → Install → Activate → Fetch
↘ Update → Reinstall → Activate
Example:
A news app uses Service Workers to cache articles for offline reading. When new articles are available, the updated service worker activates and caches the latest content.