Gestione avanzata dei Dispositivi Smart

Un dispositivo smart è un oggetto elettronico con funzionalità avanzate. Esistono diversi tipi di dispositivi smart, come **smartphone**, **smart TV** e **smartwatch**, ognuno con caratteristiche e comportamenti specifici.

Un dispositivo smart ha sempre una **marca**, un **modello** e un **anno di produzione**. A seconda della tipologia, può avere proprietà e azioni specifiche, come installare un'app per uno smartphone, regolare la luminosità per una smart TV o monitorare il battito cardiaco per uno smartwatch.

Si vuole realizzare un programma in Java che consenta di gestire un insieme di dispositivi smart, permettendo di:

- **Aggiungere e rimuovere dispositivi** da una collezione dinamica.
- **Simulare azioni tipiche di ogni dispositivo**, come installare app, aggiornare software, cambiare impostazioni di visualizzazione, ricevere notifiche.
- **Ricercare dispositivi per caratteristiche specifiche**.
- **Generare dispositivi casuali** con caratteristiche randomizzate.
- **Memorizzare dispositivi in un vettore a grandezza fissa** per gestione ottimizzata.

Attività da svolgere

1. Creare la classe base `DispositivoSmart`

La classe rappresenta un dispositivo generico con i seguenti attributi:

- **marca** (String)
- **modello** (String)
- **annoProduzione** (int)
- **batteria** (int) → percentuale di carica tra 0 e 100
- **statoAcceso** (boolean) → indica se il dispositivo è acceso o spento

Metodi da implementare:

- Un **costruttore** per inizializzare gli attributi.
- **Metodi getter e setter** per tutti gli attributi.
- **toString()** → Restituisce una rappresentazione testuale dell'oggetto.
- **accendi()** e **spegni()** → Cambiano lo stato del dispositivo.
- **caricaBatteria(int percentuale)** → Aggiunge una quantità di carica (senza superare 100).
- **scaricaBatteria(int percentuale)** → Riduce la carica (senza andare sotto 0).

---

2. Creare tre sottoclassi con funzionalità avanzate

`Smartphone` (estende `DispositivoSmart`)

Attributi aggiuntivi:

- **sistemaOperativo** (String)
- **numeroAppInstallate** (int)

Metodi aggiuntivi:

- **installaApp()** → Incrementa il numero di app installate.

- **disinstallaApp()** → Riduce il numero di app installate (minimo 0).
- **aggiornaSoftware()** → Stampa un messaggio che indica che il sistema è stato aggiornato.
- **riceviChiamata(String numero)** → Simula una chiamata in entrata.

`SmartTV` (estende `DispositivoSmart`)

Attributi aggiuntivi:

- **dimensionePollici** (int)
- **risoluzione** (String)
- **volume** (int) (tra 0 e 100)

Metodi aggiuntivi:

- **aumentaVolume()** e **diminuisciVolume()** → Modificano il livello del volume.
- **cambiaCanale(int canale)** → Stampa il nuovo canale selezionato.
- **attivaModalitàCinema()** → Simula la modalità cinema (riduce luminosità e aumenta contrasto).
- **collegaDispositivo(DispositivoSmart d)** → Simula il collegamento di un altro dispositivo (es. smartphone per il mirroring dello schermo).

`SmartWatch` (estende `DispositivoSmart`)

Attributi aggiuntivi:

- **tipoCinturino** (String)
- **autonomiaBatteria** (int, in ore)
- **battitoCardiaco** (int)

Metodi aggiuntivi:

- **monitoraBattito()** → Genera un valore casuale tra 60 e 120 bpm.
- **contaPassi()** → Restituisce un numero casuale di passi tra 1000 e 10000.
- **notificaMessaggio(String mittente, String testo)** → Simula la ricezione di una notifica.

---

3. Creare la classe `GestioneDispositivi`

Questa classe gestisce una collezione di dispositivi con un **ArrayList** e un **vettore a grandezza fissa**.

Attributi:

- **ArrayList listaDispositivi** → Contiene i dispositivi aggiunti.
- **DispositivoSmart[] archivioDispositivi** → Vettore con dimensione fissa (es. 10 posizioni).

Metodi:

- **aggiungiDispositivo(DispositivoSmart d)** → Aggiunge un dispositivo alla lista.
- **rimuoviDispositivo(int index)** → Rimuove un dispositivo dato il suo indice.
- **stampaDispositivi()** → Stampa tutti i dispositivi registrati.
- **cercaDispositivoPerMarca(String marca)** → Restituisce un elenco di dispositivi con quella marca.
- **caricaTuttiIDispositivi()** → Imposta la batteria al 100% per tutti i dispositivi.

- **generaDispositiviCasuali(int n)** → Utilizza **Random** per creare dispositivi casuali con caratteristiche diverse e li aggiunge alla lista.

---

4. Creare la classe tester `GestioneDispositiviTester.java`

Questa classe deve:

- Creare una **istanza di** `GestioneDispositivi`.
- Aggiungere dispositivi manualmente e tramite generazione casuale.
- Simulare **azioni specifiche** per ogni tipologia di dispositivo.
- Stampare il risultato delle operazioni eseguite.
