Progettate un sistema per la gestione del personale di un'azienda. L'azienda ha diverse categorie di dipendenti, tra cui:

1. **Employee**: un dipendente base con nome, cognome, data di assunzione e stipendio annuale.
2. **Manager**: un manager che eredita da Employee e ha anche un reparto di cui è responsabile e un bonus annuale.
3. **Executive**: un dirigente che eredita da Manager e ha anche un'auto aziendale e un piano di stock option.
4. **Contractor**: un consulente esterno con un compenso orario e il numero di ore lavorate al mese.
5. **Intern**: uno stagista con una durata del periodo di stage e un'indennità mensile.

Ogni categoria di dipendente deve avere i seguenti metodi:

1. **Employee**:
   - `calculateAnnualSalary()`: calcola lo stipendio annuale del dipendente.
   - `getEmployeeInfo()`: restituisce una stringa con le informazioni del dipendente (nome, cognome, data di assunzione, stipendio annuale).
2. **Manager**:
   - `calculateAnnualSalary()`: calcola lo stipendio annuale del manager, incluso il bonus.
   - `getManagerInfo()`: restituisce una stringa con le informazioni del manager (nome, cognome, data di assunzione, stipendio annuale, reparto, bonus).
3. **Executive**:
   - `calculateAnnualSalary()`: calcola lo stipendio annuale dell'executive, incluso il bonus e il valore delle stock option.
   - `getExecutiveInfo()`: restituisce una stringa con le informazioni dell'executive (nome, cognome, data di assunzione, stipendio annuale, reparto, bonus, auto aziendale, valore delle stock option).
4. **Contractor**:
   - `calculateAnnualSalary()`: calcola il compenso annuale del contractor in base al compenso orario e alle ore lavorate al mese.
   - `getContractorInfo()`: restituisce una stringa con le informazioni del contractor (nome, cognome, compenso orario, ore lavorate al mese, compenso annuale).
5. **Intern**:
   - `calculateAnnualSalary()`: calcola l'indennità annuale dello stagista in base all'indennità mensile e alla durata del periodo di stage.
   - `getInternInfo()`: restituisce una stringa con le informazioni dello stagista (nome, cognome, durata del periodo di stage, indennità mensile, indennità annuale).

Inoltre, create una classe `Company` che gestisca un elenco di dipendenti e fornisca i seguenti metodi:

- `addEmployee(Employee)`: aggiunge un dipendente all'elenco.
- `removeEmployee(Employee)`: rimuove un dipendente dall'elenco.
- `calculateTotalAnnualSalary()`: calcola la spesa salariale annuale totale dell'azienda per tutti i dipendenti.
- `displayEmployeeInfo()`: stampa le informazioni di tutti i dipendenti dell'azienda.

Infine, create un programma principale che istanzi diversi tipi di dipendenti, li aggiunga all'elenco della `Company` e testi tutti i metodi richiesti.

Richiesta dei singoli metodi:

1. `calculateAnnualSalary()`:
    - Per `Employee`, restituisca lo stipendio annuale.
    - Per `Manager`, restituisca lo stipendio annuale più il bonus annuale.
    - Per `Executive`, restituisca lo stipendio annuale più il bonus annuale e il valore delle stock option (che potrebbe essere una percentuale dello stipendio o un importo fisso).
    - Per `Contractor`, restituisca il compenso orario moltiplicato per le ore lavorate al mese e moltiplicato per 12 mesi.
    - Per `Intern`, restituisca l'indennità mensile moltiplicata per la durata del periodo di stage.
2. `getEmployeeInfo()`, `getManagerInfo()`, `getExecutiveInfo()`, `getContractorInfo()`, `getInternInfo()`:
    - Restituisca una stringa formattata con tutte le informazioni pertinenti per ogni categoria di dipendente, come specificato nella consegna.
3. `addEmployee(Employee)` e `removeEmployee(Employee)`:
    - Aggiunga o rimuova un dipendente dall'elenco della `Company`.
4. `calculateTotalAnnualSalary()`:
    - Calcoli la somma degli stipendi annuali di tutti i dipendenti dell'azienda, utilizzando il metodo `calculateAnnualSalary()` appropriato per ogni categoria di dipendente.
5. `displayEmployeeInfo()`:
    - Stampi le informazioni di tutti i dipendenti dell'azienda utilizzando i metodi `getEmployeeInfo()`, `getManagerInfo()`, `getExecutiveInfo()`, `getContractorInfo()` e `getInternInfo()` appropriati per ogni categoria di dipendente.

** **E9.11.** Realizzate una classe `Person` e due sue sottoclassi, `Student` e `Instructor`. Una persona ha un nome e un anno di nascita, uno studente ha una disciplina di specializzazione e un docente ha un salario. Per ogni classe scrivete la dichiarazione, i costruttori e il metodo `toString`. Fornite un programma di prova per collaudare classi e metodi.

- Definire una classe astratta `Persona` con i seguenti attributi e metodi:
  - Attributi: `nome` (String), `cognome` (String), `dataNascita` (LocalDate), `residenza` (String)
  - Metodi: costruttori, getter e setter, `toString()`, `equals(Object other)`, `calcolaEta()`, `isResidenteIn(String citta)`, `haLoStessoCognome(Persona p)`

- Definire una classe astratta `StudenteUniversitario` che estende `Persona` con i seguenti attributi e metodi:
  - Attributi: `matricola` (int), `annoImmatricolazione` (int), `corsoDiLaurea` (String)
  - Metodi: costruttori, getter e setter, `toString()`, `equals(Object other)`, `calcolaAnniIscrizione()`, `isImmatricolatoInCorso(String corso)`, `isStudenteFuoriCorso()`, `isStudenteLavoratore(Persona p)`

- Definire una classe `Studente` che estende `StudenteUniversitario` con i seguenti metodi aggiuntivi:
  - Metodi: `calcolaTassaIscrizione()` (restituisce un double rappresentante la tassa di iscrizione annuale), `isStudenteRegolare()`, `calcolaTassaMedia(List<Studente> studenti)`

- Definire una sottoclasse `StudenteLavoratore` di `Studente` con i seguenti attributi e metodi aggiuntivi:
  - Attributi: `azienda` (String), `stipendio` (double)
  - Metodi: costruttori, getter e setter, `toString()`, `equals(Object other)`, `calcolaRedditoAnnuale()`

- Definire una sottoclasse `StudenteFuoriSede` di `Studente` con i seguenti attributi e metodi aggiuntivi:
  - Attributi: `affitto` (double), `borsa` (double)
  - Metodi: costruttori, getter e setter, `toString()`, `equals(Object other)`, `calcolaSpeseAnnuali()`

- Nella classe `Studente`, implementare il metodo `calcolaTassaIscrizione()` che restituisce una tassa fissa di 1000 euro.
- Nella classe `StudenteLavoratore`, implementare il metodo `calcolaRedditoAnnuale()` che restituisce il reddito annuale dello studente lavoratore, moltiplicando lo stipendio mensile per 12.
- Nella classe `StudenteFuoriSede`, implementare il metodo `calcolaSpeseAnnuali()` che restituisce la somma dell'affitto annuale più la tassa di iscrizione (calcolata dal metodo `calcolaTassaIscrizione()`), meno l'eventuale borsa di studio.
- Sovrascrivere opportunamente i metodi `toString()` ed `equals(Object other)` nelle classi `Studente`, `StudenteLavoratore` e `StudenteFuoriSede`.

- Nella classe principale (main), creare alcuni oggetti di tipo `Persona`, `Studente`, `StudenteLavoratore` e `StudenteFuoriSede` e testare i metodi implementati.
- Implementare un metodo statico `confrontaEta(Persona p1, Persona p2)` nella classe `Persona` che restituisce un intero negativo se `p1` è più giovane di `p2`, zero se hanno la stessa età, un intero positivo se `p1` è più vecchio di `p2`.
- Implementare un metodo `isStudenteLavoratore(Persona p)` nella classe `StudenteUniversitario` che restituisce `true` se l'oggetto `p` è un'istanza di `StudenteLavoratore`, `false` altrimenti.
- Implementare un metodo `isStudenteFuoriCorso()` nella classe `StudenteUniversitario` che restituisce `true` se lo studente è iscritto da più di 5 anni, `false` altrimenti.
- Implementare un metodo `isStudenteRegolare()` nella classe `Studente` che restituisce `true` se lo studente ha un'età inferiore a 25 anni e non è fuori corso, `false` altrimenti.
- Implementare un metodo statico `calcolaTassaMedia(List<Studente> studenti)` nella classe `Studente` che restituisce la tassa media di iscrizione degli studenti presenti nella lista.
- Implementare un metodo `haLoStessoCognome(Persona p)` nella classe `Persona` che restituisce `true` se l'oggetto `p` ha lo stesso cognome dell'oggetto corrente, `false` altrimenti.
- Implementare un metodo `isResidenteIn(String citta)` nella classe `Persona` che restituisce `true` se la persona risiede nella città specificata, `false` altrimenti.
- Implementare un metodo `isImmatricolatoInCorso(String corso)` nella classe `StudenteUniversitario` che restituisce `true` se lo studente è iscritto al corso di laurea specificato, `false` altrimenti.