Exercise Description:

You are tasked with developing a system for managing ships and the containers they carry. The system should include different types of containers and calculate the total shipping charge based on the cargo.

Specifications:

1. **Container Class (Container.java)**:
    - This is an abstract base class that represents a container.
    - Attributes include:
        - `code`: A unique identifier for the container.
        - `destination`: The destination port of the container.
    - It should have an abstract method `calcCharge()`, which will be implemented by subclasses to calculate the charge based on container type.
2. **Bulk Container (Bulk.java)**:
    - A subclass of `Container` representing a bulk container.
    - The bulk container has an additional attribute, `weight`.
    - The shipping charge for a bulk container is calculated as `weight * 10`.
3. **Refrigerated Container (Refrigerator.java)**:
    - A subclass of `Container` representing a refrigerated container.
    - It has an additional attribute, `power`, representing the power consumption of the refrigeration unit.
    - The shipping charge is calculated as `power * 2000`.
4. **Ship Class (Ship.java)**:
    - The `Ship` class represents a ship with attributes:
        - `name`: The name of the ship.
        - `capacity`: The maximum number of containers the ship can hold.
    - It maintains a list of `Container` objects.
    - Methods include:
        - `loadContainer(Container aContainer)`: Adds a container to the ship if there's space.
        - `calcTotalCharge()`: Calculates the total charge for all containers on board by summing the charges from each container.
5. **GUI for Charge Calculation (ChargeCalculator.java & ContainerFrame.java)**:
    - A simple GUI using Swing is provided to display the total charge for a ship.
    - The `ChargeCalculator` frame contains a button that, when clicked, prints the total shipping charge for the selected ship.
    - `ContainerFrame` provides an interface for managing ships and loading containers into them.
6. **Main Class (Main.java)**:
    - The `Main` class initializes several ships and launches the GUI.

- Ships are created with names and capacities, and these ships are passed to the GUI for container management.

Task:

1. Implement the container management system as described above, with different container types (bulk and refrigerated).
2. Create a GUI to load containers onto ships and calculate the total shipping charge.
3. Ensure that the system correctly handles adding containers to ships and prevents overloading beyond the ship's capacity.
4. Display the total charge for all containers on a ship using the GUI.

Bonus:

- Extend the system by adding more types of containers or additional features in the GUI, such as selecting containers to load based on their destination.

This exercise involves concepts like inheritance, abstract classes, GUI programming with Java Swing, and container management logic.