Organism Class (Base Class) Requirements:

- Store common data attributes, such as position on the grid, whether the organism has moved in the current time step, and whether it is an ant or doodlebug.
- Include a method to move the organism on the grid. This should be a virtual function, as ants and doodlebugs will have different movement logic.
- Include a method for breeding, which will also be virtual and implemented differently by ants and doodlebugs.
- Include a method to indicate whether the organism should be removed from the grid (for example, in case it dies).

Ant Class (Derived from Organism) Requirements:

- Implement the move behavior: random move to an adjacent cell (up, down, left, or right) unless the cell is occupied or would move the ant off the grid.
- Implement the breed behavior: after surviving three time steps, create a new ant in an adjacent empty cell. The ability to breed resets every three time steps after breeding.
- Ants do not have a starve behavior, so no need to implement this method.

Doodlebug Class (Derived from Organism) Requirements:

- Implement the move behavior: first check for an adjacent cell containing an ant and move there to eat the ant. If no adjacent ant, move like an ant.
- Implement the breed behavior: breed by creating a new doodlebug in an adjacent empty cell after surviving eight time steps.
- Implement the starve behavior: if a doodlebug has not eaten an ant within three time steps, it dies and is removed from the grid.

Additional Considerations for Both Classes:

- Both classes should properly inherit from the Organism class and use its attributes and methods as appropriate.
- Both classes need to keep track of the time steps since they last bred to determine when they can breed again.
- Doodlebugs also need to keep track of the time steps since they last ate to determine if they starve.

Grid/World Class Requirements:

1. **Data Representation:**
    - Represent a 20x20 two-dimensional array or a similar data structure to hold the grid cells.
    - Each cell should hold a pointer or reference to an Organism object (Ant or Doodlebug) or be null/empty if there is no organism in the cell.
2. **Initialization:**
    - Initialize the grid with the specified number of ants and doodlebugs placed in random locations.
    - Ensure that no two organisms occupy the same cell during initialization.
3. **Movement and Position Tracking:**
    - Provide a way to move organisms within the grid, respecting the boundaries of the grid and the rule that only one organism can occupy a cell at a time.
    - Update the organism's position on the grid after each move.
4. **Breed Management:**
    - Implement logic to handle breeding of organisms, creating new ants or doodlebugs in adjacent cells when the breeding conditions are met.
5. **Starvation and Death:**
    - Handle the removal of doodlebugs from the grid if they starve.
    - Remove any organism from the grid if necessary (for example, when an ant is eaten by a doodlebug).
6. **Turn Management:**
    - Ensure that all doodlebugs move before ants in each turn.
    - Keep track of whether an organism has moved in a time step to prevent multiple moves.
7. **Drawing the Grid:**
    - Provide a method to output the grid to the console using ASCII characters: "o" for ants, "X" for doodlebugs, and "-" for empty cells.

- Update the display after each time step to reflect the new state of the grid.

8. **Simulation Control:**
    - Include a method to advance the simulation by one time step, updating the position of the organisms, handling breeding and starvation, and updating the grid display.
    - Potentially include a way for the simulation to respond to user input to advance the simulation steps.
9. **Consistency Checks:**
    - Implement checks to prevent organisms from moving off the grid.
    - Ensure the grid correctly reflects the state of the simulation at the end of each time step.