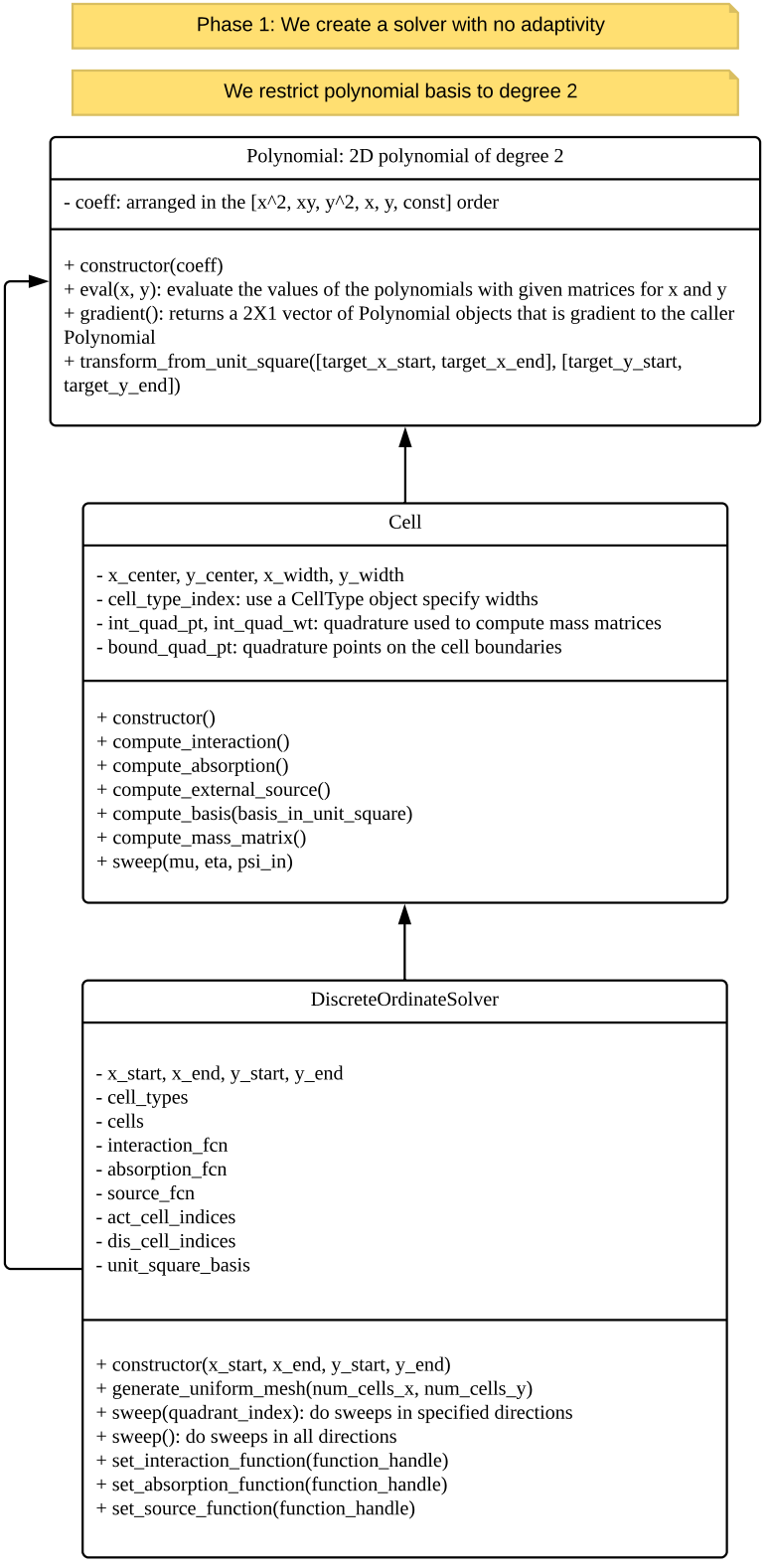
Phase 1: We create a solver with no adaptivity
We restrict polynomial basis to degree 2
Polynomial: 2D polynomial of degree 2
- coeff: arranged in the [x^2, xy, y^2, x, y, const] order
+ constructor(coeff)
+ eval(x, y): evaluate the values of the polynomials with given matrices for x and y
+ gradient(): returns a 2X1 vector of Polynomial objects that is gradient to the caller Polynomial
+ transform_from_unit_square([target_x_start, target_x_end], [target_y_start, target_y_end])
Cell
- x_center, y_center, x_width, y_width
- cell_type_index: use a CellType object specify widths
- int_quad_pt, int_quad_wt: quadrature used to compute mass matrices
- bound_quad_pt: quadrature points on the cell boundaries
+ constructor()
+ compute_interaction()
+ compute_absorption()
+ compute_external_source()
+ compute_basis(basis_in_unit_square)
+ compute_mass_matrix()
+ sweep(mu, eta, psi_in)
DiscreteOrdinateSolver
- x_start, x_end, y_start, y_end
- cell_types
- cells
- interaction_fcn
- absorption_fcn
- source_fcn
- act_cell_indices
- dis_cell_indices
- unit_square_basis
+ constructor(x_start, x_end, y_start, y_end)
+ generate_uniform_mesh(num_cells_x, num_cells_y)
+ sweep(quadrant_index): do sweeps in specified directions
+ sweep(): do sweeps in all directions
+ set_interaction_function(function_handle)
+ set_absorption_function(function_handle)
+ set_source_function(function_handle)