**package** level;

**class** ArrayQueue **implements** Queue {

**public** **int** front;

**public** **int** rear;

**public** **int** count;

**public** **int** queueSize;

**public** **int** increment;

**public** Object[] itemArray;

**public** ArrayQueue()

{

front = 0;

rear = 0;

count = 0;

queueSize = 50;

increment = 10;

itemArray = **new** Object[queueSize];

}

**public** **boolean** isEmpty() {

**return** (count == 0);

}

**public** **void** enqueue(Object x) //(1) 큐 삽입 메소드

{

**if** (count == queueSize) // 아이템의 갯수가 큐 사이즈와 같으면

queueFull(); // queueFull() 메소드 호출

itemArray[rear] = x; // rear는 큐의 꼬리(마지막)을 뜻함, 배열 마지막 칸에 x(아이템)를 삽입

rear = (rear + 1) % queueSize; // 큐의 꼬리를 증가 시켜줌

count++; // 아이템의 갯수 증가

}

**public** **void** queueFull() //(2) 큐 포화 메소드

{

**int** oldsize = queueSize; // 현재 큐 사이즈를 기억

queueSize += increment; // 큐사이즈를 증가값(10)만큼 증가

Object[] tempArray = **new** Object[queueSize]; // 증가한 사이즈 만큼 새로운 배열을 선언

**for** (**int** i = 0; i < count; i++) // 새로운 배열에 원래 배열을 옮기는 반복문

{

tempArray[i] = itemArray[front]; // 배열의 머리(front)를 새로운 배열에 넣어줌

front = (front + 1) % oldsize; // 머리(front)를 이동시켜줌

}

itemArray = tempArray; // 배열을 원래 배열로 이동시켜줌

front = 0; // 머리(front)를 초기화 시킴으로써 맨 앞을 가리키게 함

rear = count; // 꼬리(rear)를 아이템의 갯수만큼, 맨 끝으로 가게 함

}

**public** Object dequeue() //(3) 큐 삭제 메소드

{

**if** (isEmpty()) // 큐가 비어 있으면

**return** **null**; // null 값 반환

Object item = itemArray[front]; // 아이템 변수에 현재 큐에 있는 맨 첫 원소를 담아줌

front = (front + 1) % queueSize; // 머리(front) 값을 증가시킴

count--; // 아이템 수 감소

**return** item; // 아이템 반환

}

}

20135185\_황석진\_트리예상구조

