Java Desktop Application Development
CST 3613
Fall 2022

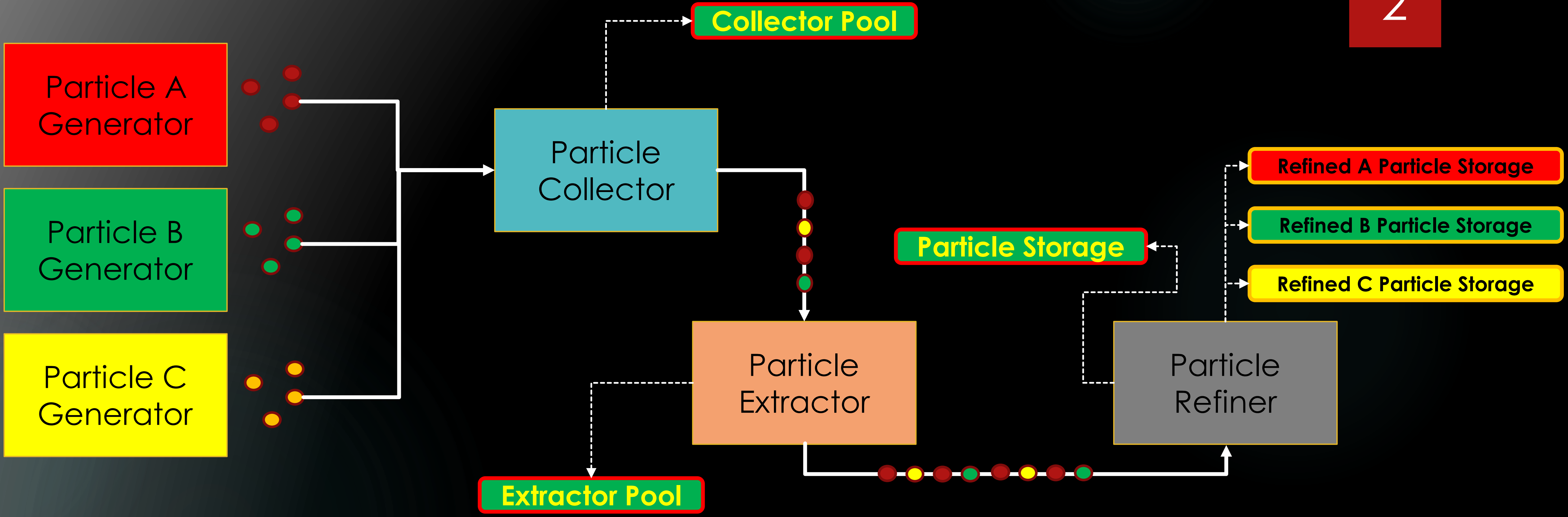
OOP

Practice Midterm Exam

COMPREHENSIVE OOP EXERCISE

Professor HG Locklear

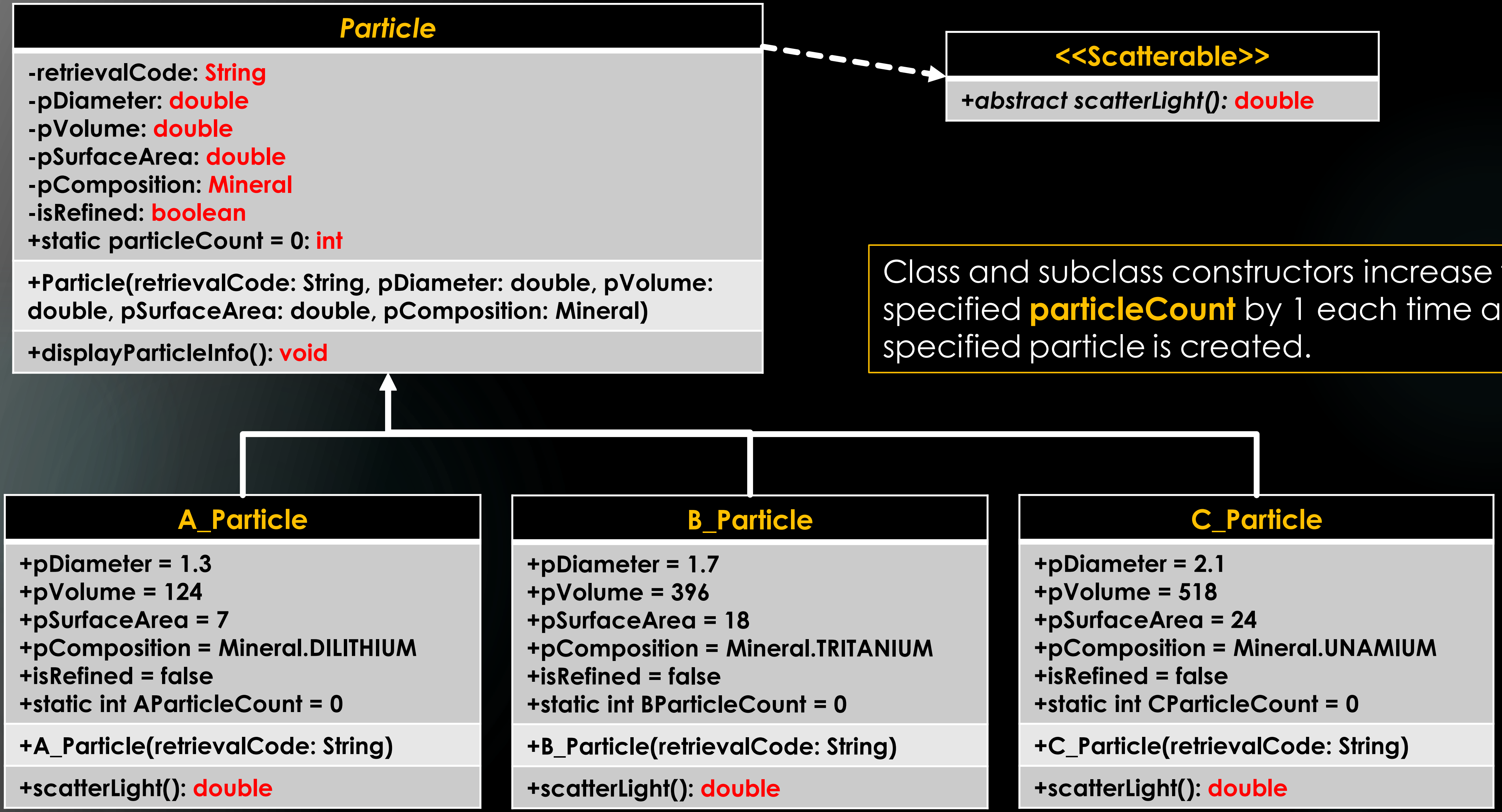
General



Particle Processing consists of 6 components and processes:

1. Particle Generator generate a specified particle type.
2. Particle Collectors collect the generated particles into a Collector Pool of particles.
3. Particle Extractor extracts particles from the Collector Pool of the Particle Collector its Extractor Pool.
4. Particle Refiner empties the Extractor Pool of the Particle Extractor into its Particle Storage and refines each particle ultimately storing them in a segregated particle storage containers.

Particle Class



Particle

-retrievalCode: String
-pDiameter: double
-pVolume: double
-pSurfaceArea: double
-pComposition: Mineral
-isRefined: boolean
+static particleCount = 0: int

+Particle(retrievalCode: String, pDiameter: double, pVolume: double, pSurfaceArea: double, pComposition: Mineral)

+displayParticleInfo(): void

<<Scatterable>>

+abstract scatterLight(): double

Class and subclass constructors increase the specified **particleCount** by 1 each time a specified particle is created.

A_Particle

+pDiameter = 1.3
+pVolume = 124
+pSurfaceArea = 7
+pComposition = Mineral.DILITHIUM
+isRefined = false
+static int AParticleCount = 0

+A_Particle(retrievalCode: String)

+scatterLight(): double

B_Particle

+pDiameter = 1.7
+pVolume = 396
+pSurfaceArea = 18
+pComposition = Mineral.TRITANIUM
+isRefined = false
+static int BParticleCount = 0

+B_Particle(retrievalCode: String)

+scatterLight(): double

C_Particle

+pDiameter = 2.1
+pVolume = 518
+pSurfaceArea = 24
+pComposition = Mineral.UNAMIUM
+isRefined = false
+static int CParticleCount = 0

+C_Particle(retrievalCode: String)

+scatterLight(): double

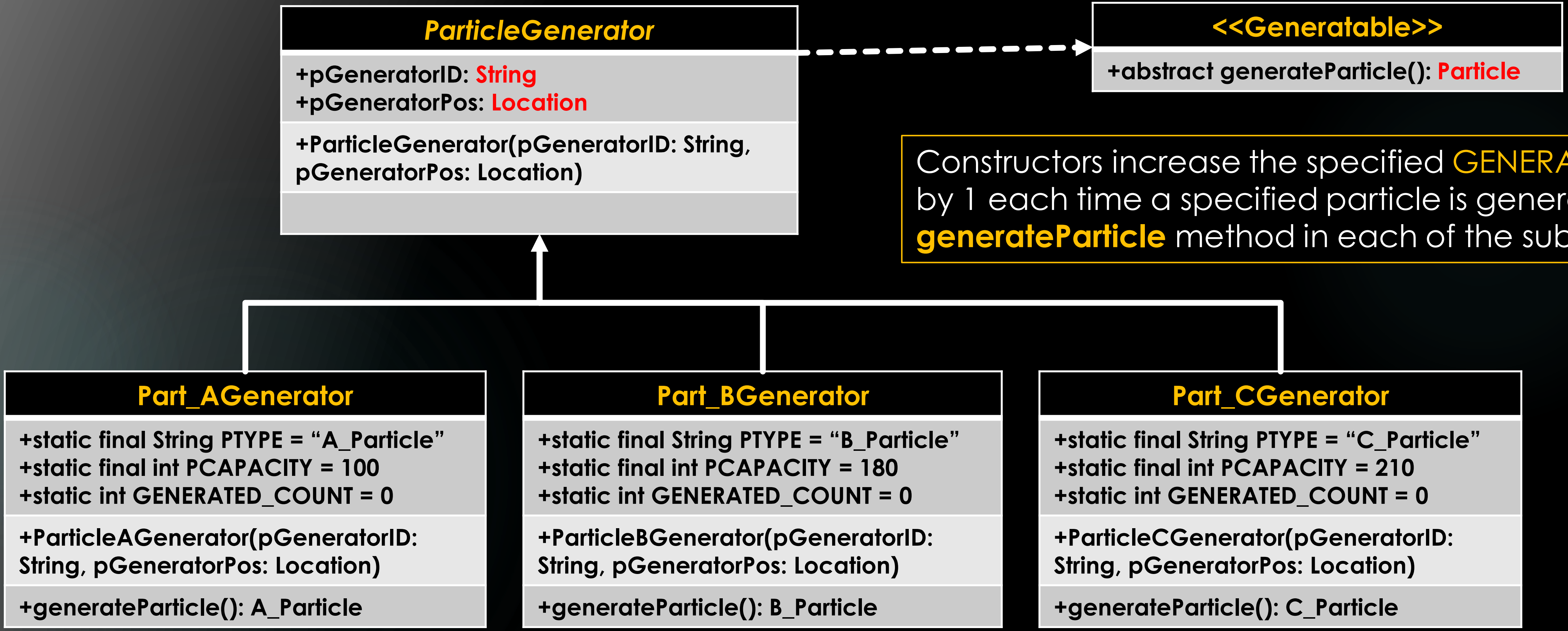
Particle Class and Subclass Methods

displayParticleInfo		
Purpose	Display Particle Information	
Input	**Processing**	**Output**
None	Displays all information about a Particle object in the specified format	void …formatted output to console

You can specify the format

scatterlight		
Purpose	Calculate Particle Light Scattering Value	
Input	**Processing**	**Output**
None	Calculates the light scattering effects value of the Particle object based on the following formulae A_Particles -> Mineral Strength * SQRT(10) * 0.28 * Mineral Mass B_Particles -> Mineral Strength * SQRT(10) C_Particles -> Mineral Mass * 10	double…light scattering value

ParticleGenerator Class


ParticleGenerator
+pGeneratorID: String
+pGeneratorPos: Location
+ParticleGenerator(pGeneratorID: String, pGeneratorPos: Location)
<<Generatable>>
+abstract generateParticle(): Particle
Part_AGenerator
+static final String PTYPE = "A_Particle"
+static final int PCAPACITY = 100
+static int GENERATED_COUNT = 0
+ParticleAGenerator(pGeneratorID: String, pGeneratorPos: Location)
+generateParticle(): A_Particle
Part_BGenerator
+static final String PTYPE = "B_Particle"
+static final int PCAPACITY = 180
+static int GENERATED_COUNT = 0
+ParticleBGenerator(pGeneratorID: String, pGeneratorPos: Location)
+generateParticle(): B_Particle
Part_CGenerator
+static final String PTYPE = "C_Particle"
+static final int PCAPACITY = 210
+static int GENERATED_COUNT = 0
+ParticleCGenerator(pGeneratorID: String, pGeneratorPos: Location)
+generateParticle(): C_Particle

Constructors increase the specified GENERATED_COUNT by 1 each time a specified particle is generated by the **generateParticle** method in each of the subclasses.

The **retrievalCode** for any Particle object is determined by:
A_Particle -> "A" + GENERATED_COUNT, B_Particle -> "B" + GENERATED_COUNT, C_Particle -> "C" + GENERATED_COUNT

ParticleGenerator Method

generateParticle		
Purpose	Creates Particles	
Input	Processing	Output
NONE	Creates a new Particle object (of the specified type). A_Particle retrievalCode = “A” + GENERATED_COUNT B_Particle retrievalCode = “B” + GENERATED_COUNT C_Particle retrievalCode = “C” + GENERATED_COUNT	Particle object

ParticleCollector Class

ParticleCollector
-collectorID: String -collectorPOS: Location -collectorPool: ArrayList<Particle> +static final int COLLECTOR_CAPACITY = 500
+ParticleCollector(collectorID: String, collectorPOS: Location)
+collectoParticle(Particle p): boolean

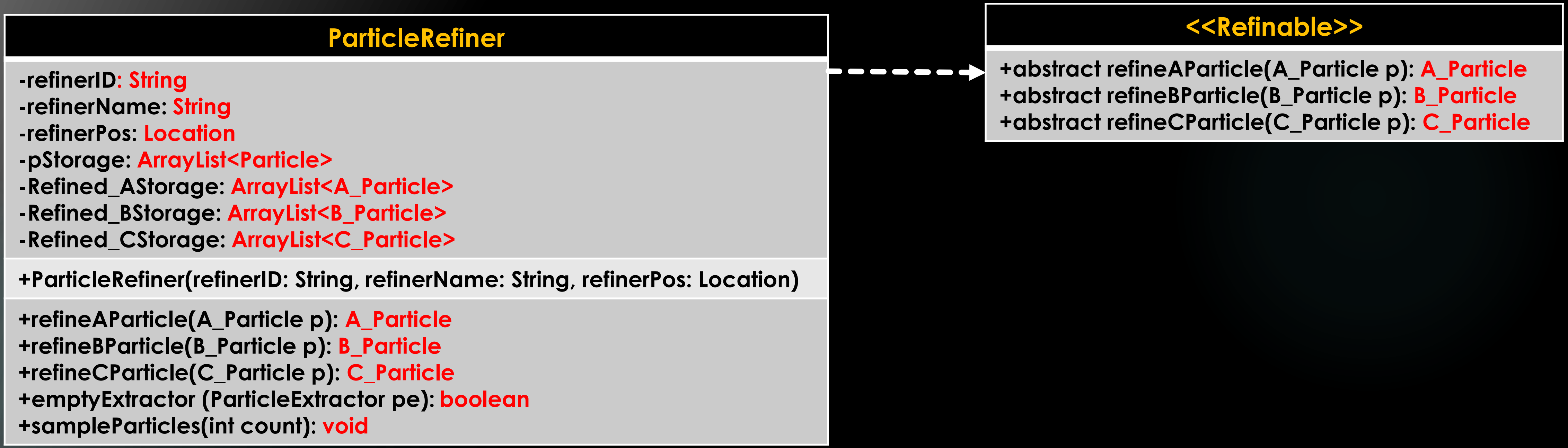
collectParticle		
Purpose	Adds Particles to the Particle Collector	
Input	Processing	Output
Particle object	Adds a Particle object to the collectorPool of this ParticleCollector if the addition of this particle does not exceed the COLLECTOR_CAPACITY	boolean…returns true if the operation is successful and false otherwise.

ParticleExtractor Classes

ParticleExtractor
-extractorID: String -extractorPOS: Location -extractorPool: ArrayList<Particle> +static final int EXTRACTOR_CAPACITY = 500
+ParticleExtractor(extractorID: String, extractorPOS: Location)
+extractParticles(ParticleCollector pc): boolean

extractParticles		
Purpose	Extracts Particles from the Particle Collector	
Input	Processing	Output
ParticleCollector object	Adds all the Particle objects from the ParticleCollector collectorPool to this ParticleExtractor's extractorPool if the addition of the Particle objects does not exceed the EXTRACTOR_CAPACITY of the ParticleExtractor	boolean…returns true if the operation is successful and false otherwise. If operation is unsuccessful displays '*Extractor Pool does not have sufficient capacity*' to the console.

ParticleRefiner Class



ParticleRefiner
-refinerID: String
-refinerName: String
-refinerPos: Location
-pStorage: ArrayList<Particle>
-Refined_AStorage: ArrayList<A_Particle>
-Refined_BStorage: ArrayList<B_Particle>
-Refined_CStorage: ArrayList<C_Particle>
+ParticleRefiner(refinerID: String, refinerName: String, refinerPos: Location)
+refineAParticle(A_Particle p): A_Particle
+refineBParticle(B_Particle p): B_Particle
+refineCParticle(C_Particle p): C_Particle
+emptyExtractor (ParticleExtractor pe): boolean
+sampleParticles(int count): void
<<Refinable>>
+abstract refineAParticle(A_Particle p): A_Particle
+abstract refineBParticle(B_Particle p): B_Particle
+abstract refineCParticle(C_Particle p): C_Particle

ParticleRefiner Methods

refineAParticles		
Purpose	Refines Particles	
Input	Processing	Output
A_Particle object	Sets the isRefined attribute to true for the Particle object	A_Particle

Create similar versions for B_Particles and C_Particles

emptyExtractor		
Purpose	Removes Particles from the Particle Extractor	
Input	Processing	Output
ParticleExtractor object	Adds all the Particle objects from the ParticleExtractor object to this pStorage and then separates them into the appropriate Refined_AStorage, Refined_BStorage, or Refined_CStorage	boolean…returns true if the operation is successful and false otherwise.

sampleParticles		
Purpose	Displays Information about Particles in the Particle Refiner Particle Storage	
Input	Processing	Output
Number of Particle objects to be displayed	Calls the displayParticleInfo method for the specified number of particles in the pStorage of this ParticleRefiner	void…formatted output to console

ParticleRefiner Methods

displayInfo		
Purpose	Displays Information about the Particle Refiner	
Input	Processing	Output
NONE	Displays all information about a ParticleRefiner in the specified format	void…formatted output to console

You can specify the format

Supporting Classes/Enumerations

Location
-name: String -X: int -Y: int
Location(name: String, x: int, y: int)
+euclideanDistance(Location L): double +toString(): String

ENUMERATION Mineral

DILITHIUM(40,140)
TRITANIUM(80,180)
UNAMIUM(90,270)
-mass: double
-strength: double

-Mineral(mass: double, strength: double)

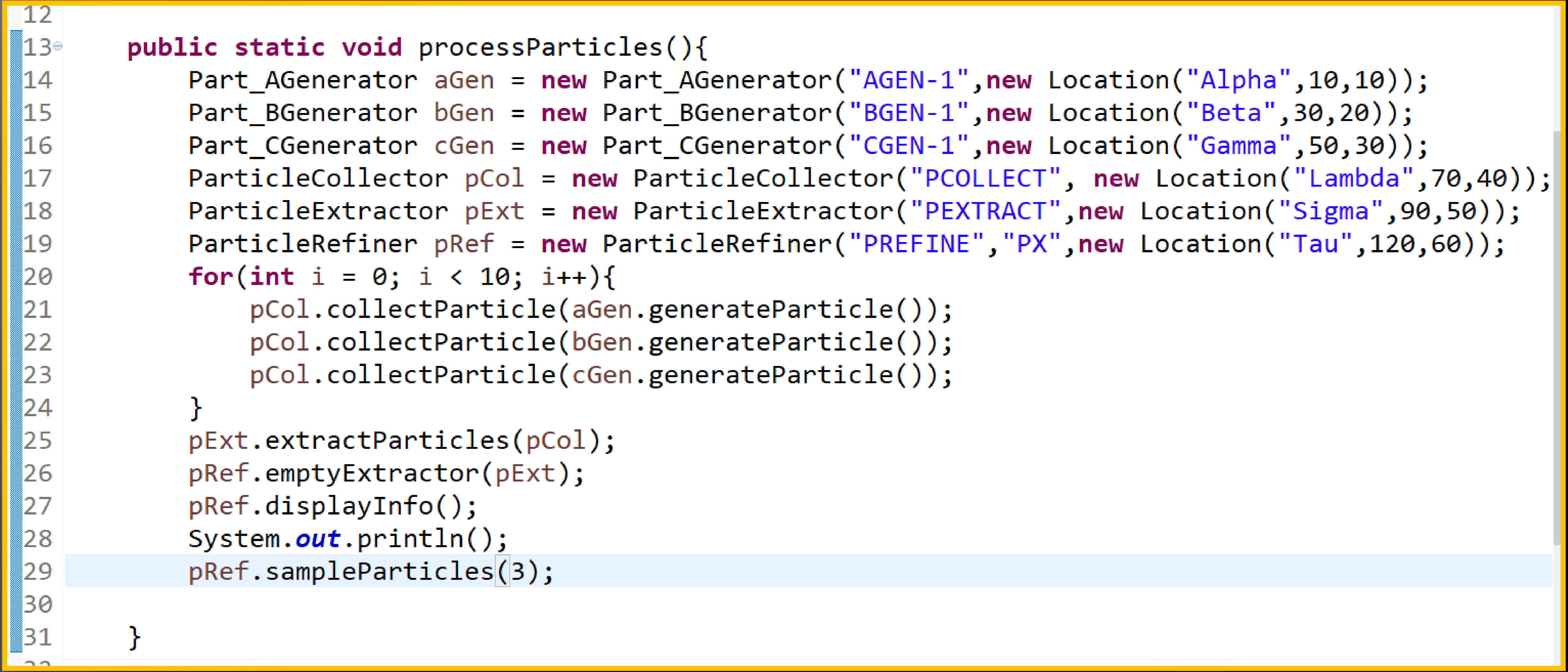
ParticleProcessing
+static main(String[ ] args): void +processParticles(): void

See Slide 15

euclideanDistance		
Purpose	Calculates Euclidean Distance between two Locations	
Input	Processing	Output
Location object	Calculates the Euclidean distance between this Location object and the specified Location object	double…Euclidean distance between Location objects

Program Testing

Utilize this method to test your program for correctness

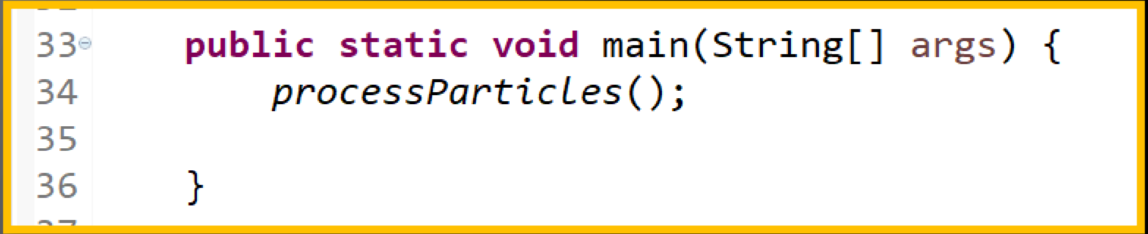


```java
    public static void processParticles(){
        Part_AGenerator aGen = new Part_AGenerator("AGEN-1",new Location("Alpha",10,10));
        Part_BGenerator bGen = new Part_BGenerator("BGEN-1",new Location("Beta",30,20));
        Part_CGenerator cGen = new Part_CGenerator("CGEN-1",new Location("Gamma",50,30));
        ParticleCollector pCol = new ParticleCollector("PCOLLECT", new Location("Lambda",70,40));
        ParticleExtractor pExt = new ParticleExtractor("PEXTRACT",new Location("Sigma",90,50));
        ParticleRefiner pRef = new ParticleRefiner("PREFINE","PX",new Location("Tau",120,60));
        for(int i = 0; i < 10; i++){
            pCol.collectParticle(aGen.generateParticle());
            pCol.collectParticle(bGen.generateParticle());
            pCol.collectParticle(cGen.generateParticle());
        }
        pExt.extractParticles(pCol);
        pRef.emptyExtractor(pExt);
        pRef.displayInfo();
        System.out.println();
        pRef.sampleParticles(3);

    }
```

Program Testing

Call the **processParticles** method in the main method of your program.



```
    public static void main(String[] args) {
        processParticles();

    }
```

The main method of your program **can contain only** the call of the processParticles method.