Design Rationale for Requirement 1:

To fulfill requirement 1, I designed the Scraps class by extending it from the Item class and setting its portability attribute to true. This design choice allows for the creation of instances such as LargeBolt and MetalSheet by utilizing different parameters. As these instances don't exhibit distinct differences beyond their parameters, it is logical to initialize them through the Scraps class instead of creating separate classes for each item (adhering to the Don't Repeat Yourself principle).

Moreover, this design aligns with the Dependency Inversion Principle, as Scraps extends an abstract class. This principle asserts that details should depend on abstractions. By extending Scraps from Item, rather than creating another class extending GameEntity, the design follows this principle.

Furthermore, the design upholds the Open-Closed Principle. Methods utilizing this type of GameEntity can specify the parameter type as Item. Consequently, when adding new types of scraps, there's no need to modify the code of those methods. This adherence to the Open-Closed Principle ensures that the code remains open for extension but closed for modification.