Class Diagram for RPG Platform

This diagram illustrates the main classes, their attributes and methods, and the relationships between them, such as inheritance and association.

```
classDiagram
    direction LR

    class Character {
        <<Abstract>>
        -name: String
        -healthPoints: int
        -strengthPoints: int
        -money: double
        +move(Location)
        +pickUp(Item)
        +drop(Item)
        +acquireSkill(Skill)
        +interact(Character, Interaction)
        +die()
    }

    class PlayerCharacter {
        +getInput()
    }

    class AICharacter {
        +calculateNextMove()
    }

    class MagicalCharacter
    class NonMagicalCharacter
    class Wizard
    class Elf
    class Human
    class Dwarf
```

```
class Item {
   <<Abstract>>
   -name: String
}

class Weapon
class Container
class Consumable {
   +consume(Character)
}

class Sword
class Axe
class Backpack
class Potion
class Bread

class Skill {
   -name: String
   -description: String
}

class MagicalSkill {
   -manaCost: int
}

class Location {
   -terrainType: String
   -traversalCost: int
   -money: double
}

class Interaction {
   <<Interface>>
   +execute(Character, Character)
}
```

```
class Combat
class Trading

%% --- INHERITANCE / GENERALIZATION ---
Character <|-- PlayerCharacter
Character <|-- AICharacter
Character <|-- MagicalCharacter
Character <|-- NonMagicalCharacter
MagicalCharacter <|-- Wizard
MagicalCharacter <|-- Elf
NonMagicalCharacter <|-- Human
NonMagicalCharacter <|-- Dwarf

Item <|-- Weapon
Item <|-- Container
Item <|-- Consumable
Weapon <|-- Sword
Weapon <|-- Axe
Container <|-- Backpack
Consumable <|-- Potion
Consumable <|-- Bread

Skill <|-- MagicalSkill

Interaction <|.. Combat
Interaction <|.. Trading

%% --- ASSOCIATIONS & AGGREGATIONS ---
Character "1" o-- "0..*" Skill : has >
Character "1" o-- "0..2" Item : carries >
Character "*" -- "1" Location : is at

Container "1" o-- "0..*" Item : contains >
Location "1" o-- "0..*" Item : contains >
Location "1" -- "0..6" Location : is neighbor to

Character "2..*" -- "1" Interaction : participates in
```

Design Explanation

Here's a breakdown of the key design choices reflected in the diagram:

* Abstract Classes: Character and Item are marked as abstract classes. This is because you wouldn't create a generic "Character" or "Item" in the game; you would create a specific type, like a Wizard (which is a MagicalCharacter) or a Sword (which is a Weapon). This enforces a strong, extensible structure.

* Inheritance Hierarchies:
  * [cite_start]Character: The Character class is specialized into PlayerCharacter and AICharacter to separate control logic. [cite_start]It's also specialized for the "Wizzo" game into MagicalCharacter and NonMagicalCharacter, with further specializations like Wizard and Human.
    * [cite_start]Item: The Item class is extended by Weapon, Container, and Consumable, each representing a major item category with unique behaviors.
  * [cite_start]Skill: A MagicalSkill is a special type of Skill, representing the rule that some skills are magical.

* Interfaces: Interaction is designed as an interface. [cite_start]This defines a contract that any type of interaction (like Combat or Trading) must adhere to, ensuring they all have an execute() method. This makes the system flexible enough to add new interaction types in the future.

* Relationships and Multiplicity:
  * Aggregation (hollow diamond o--): This "has-a" relationship is used where objects are part of another but can exist independently.
    * [cite_start]A Character has Skills and carries Items.
    * [cite_start]A Container (like a backpack) contains other Items.
    * [cite_start]A Location can contain loose Items and money on the ground.
  * Association (--): This represents a general relationship between classes.
    * [cite_start]Many (*) Characters can be at one (1) Location.
      * [cite_start]A Location is connected to up to six neighboring Locations, representing the hexagonal world map.