Introduction to Classes and Objects in Java

A class in Java serves as a model or blueprint for building objects. It specifies the data and behavior (methods) of the objects derived from the class. Objects, with their own collection of data (known as instance variables) and methods, represent a particular instance of a class.

Defining a Class

In Java, the class keyword is used in conjunction with the class name to define a class. The following components should be part of the class definition:

1. Instance variables: These are variables that depict the data or state of an object. They are also known as fields or attributes.
2. Constructors: These unique methods are employed in the creation and initialization of objects. They have identical names to the class and lack a return type.
3. Methods: These are functions that define the behavior of an object. They may accept arguments and have a return type.

Here is an example of a straightforward Java class definition:

```java
public class Student {
  private String name;
  private int age;

  public Student(String name, int age) {
    this.name = name;
    this.age = age;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public int getAge() {
    return age;
  }

  public void setAge(int age) {
    this.age = age;
  }

  public void printDetails() {
    System.out.println("Name: " + name + ", Age: " + age);
  }
}
```

Name and age are two attributes that belong to the Student class. The two arguments it accepts in its constructor, are used to initialize the instance variables. Four additional methods exist:

getName(), setName(), getAge(), and setAge(). Because you can use these methods to access (get) or modify (set) the value of the instance variables, they are known as accessor and mutator methods. The Student object's information is printed using the printDetails() method.

Creating Objects

Using the new keyword, the class name, and the constructor with the necessary arguments, we can create an object from a class. An example of how to create an object of the Student class is provided below:

```
4    Student s1 = new Student("John", 20);
```

This generates an object of the Student class named "John" who is 20 years old. Then, you can access, change, or perform actions on the object's data using its methods. For instance:

```
4    Student s1 = new Student("John", 20);
5    s1.setName("Jane");
6    s1.setAge(22);
7    s1.printDetails(); // prints "Name: Jane, Age: 22"
```

Conclusion

In this overview, we covered Java classes and objects. We learned how to define classes by using instance variables and constructors. We also learned how to create objects as instances of these classes. By using methods including accessors and mutators, we can read and change the data of objects as well as perform other actions.

Blackjack Game Outline

1. Go to the file Card.java
    a. Create the constructor for a card with a suit and a value
    b. Create the Card.getSuit() method which returns the suit of the card
    c. Create the Card.getSuit() method which returns the value of the card
2. Go to the file Deck.java
    a. Create the constructor for a deck of 52 cards
        i. Create an empty ArrayList of cards
        ii. Create a String array with the four suits
        iii. Create a String array with the thirteen values
        iv. Use a nested for loop to add all possible cards to the empty ArrayList
    b. Complete the Deck.shuffle() method by adding code similar to the code in the constructor to create a new Deck of 52 cards
    c. Create the Deck.deal() method which uses the ArrayList.remove() method to remove a card from the deck and return the removed card
3. Go to the file Hand.java
    a. Create the constructor for an empty hand
    b. Create the Hand.addCard() method which uses the ArrayList.add() method to add a given card to the hand
    c. Create the Hand.getCards() method which returns the list of cards in the hand
    d. Complete the Hand.getTotalValue() method which loops through each card in the Hand to calculate the total value
        i. If a card is an ace, increment the total and ace variables by 1
        ii. If a card is a face card, increment the total by 10
        iii. If a card is not an ace or a face card, increment the total variable by the number on the card using the Integer.parseInt() method
        iv. For each ace in the hand, increment the total by 10 repeatedly as long as the total does not go over 21
4. Go to the file Game.java
    a. Create the constructor for a blackjack game which creates a new deck and empty hands for the player and dealer
    b. Complete the Game.play() method which plays a game of blackjack
        i. Shuffle the deck and deal two cards to each player
        ii. Code the events of the player's turn, which displays the hands of both players, prompts the player to choose an action, and acts accordingly
        iii. Code the events of the dealer's turn, which deals a card to the dealer if they don't have at least 17 points
        iv. Compare the scores of the player and the dealer, printing messages based on the outcome of the game
5. Go to the file Main.java
    a. In the infinite loop, create a new Game object and call the Game.play() method
    b. At the end of the loop, ask the player if they want to play again, breaking out of the loop if they want to stop