1.

(a)

AbstractBankAccount[] myAccounts = new AbstractBankAccount[10]; // or however many accounts you need

(b)

public void displayAccountDetails(AbstractBankAccount account) {

textArea.setText(account.toString());

}

©

textArea.setText(account.toString());

(d)

displayAccountDetails(myAccounts[0]); // Passing the first account object

2.

An abstract class can be used when you want to provide some common base functionality or state (fields) to subclasses, while still enforcing that certain methods must be implemented by the subclasses. Unlike interfaces, abstract classes can contain fields and concrete (non-abstract) methods that can be shared among all subclasses.

3.

public abstract class Bike {

}

4.

Bike bike4 = new Bike();

System.out.println(bike4.toString());

5.

public String toString() {

return "Description of the bike";

}

AFTER:-

System.out.println(bike1.toString());

6.

public class Animal {

public void makeNoise() {

System.out.println("talk");

}

}

public class Dog extends Animal {

public void makeNoise() {

System.out.println("Bark");

}

}

OUTPUT ANALYSIS:-

Animal animal = new Animal();

animal.makeNoise();

Dog dog = new Dog();

dog.makeNoise();

Animal animaldog = new Dog();

animaldog.makeNoise();