# Advanced Programming

## Object Oriented Programming in Java-III

Computer Science

# Object Oriented Key Principles

1. Data Abstraction and Encapsulation
2. Inheritance
3. Polymorphism

# Polymorphism

- Polymorphism - The ability of a superclass variable to behave as a subclass variable
- How?
  - Subclasses inherit all public and protected fields
  - Thus these can be accessed by a superclass variable
- Limitations!
  - Superclass cannot access new fields and methods within sub classes
  - However if we apply a cast, then the cast object reference can access all the new fields and methods

```
CurrentAccount acc1 = new
CurrentAccount("100003", "Mr Nasir", 100));

SavingsAccount acc2 = new
SavingsAccount("100005", "Mr Saeed", 5.0));

BankAccount ba1,ba2;

ba1= acc1;
ba2= acc2;

system.out.println(ba1.getBalance());
system.out.println(ba2.getBalance());
ba1.debit(200);
```

## Accessing inherited methods

- When we call a method with multiple versions (overrides)
  - JVM will use the method version which corresponds to the *actual* object type
    - Not the reference type
  - I.e. the overridden version will be used if appropriate

## Accessing new methods

- When a new (not overridden) method has to be accessed we have to apply a cast
  - Casting to a CurrentAccount object

```
(CurrentAccount)ba1.setOverdraftLimit(amount);
```

  - Casting to a SavingsAccount object

```
(SavingsAccount)ba2.addInterest();
```

- The cast allows the sub class object to overrule the super class reference
- Problem:
  - Applying incorrect sub class will generate an exception!
  - How can we tell which element should be cast?

# instanceof Keyword

- Checking for `CurrentAccount` subclass

```
if ( ba1 instanceof CurrentAccount ){ }
```

- Checking for `SavingsAccount` subclass

```
if ( ba2 instanceof SavingsAccount ){ }
```

- Generally, use `instanceof` rarely

## Object Oriented Key Principles

- Data Abstraction and Encapsulation
  - Internal information is hidden and a well defined interface provides access to allowed information
- Inheritance
  - A subclass can *inherit* the fields and methods of a superclass
- Polymorphism
  - A subclass object can be treated as a superclass

## Benefits of Using Objects

- Modularity
  - The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.
- Information-hiding
  - By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- Code re-use
  - If an object already exists , you can use that object in your program.
- Pluggability and debugging ease
  - If a particular object turns out to be problematic, you can simply remove it from your application.