Time	Narration
00:01	به spoken tutorial در Polymorphism در Java خوش آمدید.
00:06	در این برنامه این موارد را یاد می گیریم: Polymorphism در Java و Run-time polymorphism Compile-time polymorphism
00:19	در اینجا ما از Ubuntu Linux Version 12.04 و JDK Version 1.7 Eclipse 4.3.1 استفاده می کنیم.
00:31	برای این برنامه شما باید با Java و Eclipse IDE آشنایی داشته باشید.
00:37	همچنین شما باید با Subclassing و Method overriding و overloading آشنایی داشته باشید.
00:43	اگر نه برای برنامه‌های Java مربوطه به وب سایت ما مراجعه کنید.
00:48	Polymorphism توانایی یک object برای گرفتن فرمهای مختلف می باشد.
00:54	فواید مهم Polymorphism عبارتند از : پیچیدگی را کم می‌کند و دوباره استفاده کردن کد
01:03	در java دو نوع Polymorphism وجود دارد: Compile-time و Run-time Polymorphism
01:11	Compile-time Polymorphism که لزوماً به Method overloading اشاره می‌کند که به آن Static Binding نیز گفته می شود.
01:20	Run-time Polymorphism لزوماً به Method overriding اشاره می‌کند و به آن Dynamic Binding نیز گفته می شود.

01:29	ما از قبل Run-time polymorphism که Method overriding است را یاد گرفته ایم.
01:35	به eclipse IDE می رویم. من در برنامه قبل پروژه با نام MyProject را ایجاد کرده ام.
01:44	ما فایل کدها از برنامه Using final keyword را استفاده می کنیم.
01:49	Employee class که parent class می باشد.
01:52	Manager class که subclass است.
01:55	Manager class شامل یک متغیر دیگر department می باشد.
02:01	متد برای Manager class که ()getDetails است و متد برای Employee class که ()getDetails است را overrides می کند.
02:08	ما getDetails() method را با object از Manager class که Manager است فرا می خوانیم.
02:16	برای پرینت جزئیات system.out.println Details of Manager Class را تایپ کنید.
02:28	برنامه را save و run کنید. پس ما مقدار متغیر department را در خروجی می بینیم.
02:37	پس subclass method در runtime فراخوانی می شود.
02:42	method invocation با JVM تعیین می‌شود نه کامپایلر
02:48	بنابراین به این Runtime polymorphism یا method overriding گفته می شود.
02:55	ما یاد گرفتیم که Run time polymorphism چیست.
02:58	حالا Virtual Method Invocation را یاد می گیریم.
03:03	به Employee class در Eclipse IDE بیایید.
03:07	کلیدواژه های static و final را برای متغیر name حذف کنید.
03:13	و setName method را uncomment کنید.
03:16	و static block را حذف کنید. فایل را save کنید.
03:21	به TestEmployee class بیایید. مقدار instance از manager.setName(“Nikkita; (”Dinesh را uncomment کنید.
03:31	ما این instance را uncomment می‌کنیم چون ما ()method setName در Employee class را uncomment کرده ایم.
03:38	حالا Employee object emp1 برای مرجع Employee class را پیدا می کنیم.
03:46	Employee emp1 = new Employee open and close parenthesis semicolon را تایپ کنید.
03:57	مقادیر setEmail و setName برای Employee را شروع می کنیم.

04:03	و emp1.setName(“Jayesh”); emp1.setemail(“pqr@gmail.com”); را تایپ کنید.
04:16	برای پرینت جزئیات کارکنان این را تایپ کنید: system.out.println(“Details of Employee class:” emp1,getDetails()) semicolon
04:37	Manager object emp2 برای مرجع Employee class را پیدا کنید. Employee emp2 = new Manager open and close parenthesis semicolon را تایپ کنید.
04:54	ما قادر به انجام این کار هستیم چون هر Java object که بیشتر از یک IS-A تست را قبول شود polymorphic می باشد.
05:04	در Java همه objects ها polymorphic هستند چون هر object تست IS-A را برای نوع خود و کلاس object پاس می کند.
05:16	A Manager IS-A Employee A Manager IS-A Manager A Manager IS-A Object
05:23	تنها راه دسترسی به object از طریق reference variable می باشد.
05:29	Reference variables مثل emp1, emp2 و manager
05:36	در اینجا ما دو Manager objects را پیدا می کنیم: یکی با مرجع Employee class و دیگری با مرجع manager class
05:47	با استفاده از emp2 object مقادیر برای setEmail, setName و setDepartment را شروع می کنیم.
05:55	این را تایپ کنید: emp2,setName(“Ankita”); emp2.setEmail(“xyz@gmail.com”); emp2.setDepartment(“IT”);

06:14	می‌بینیم که یک اشتباه است که method setDepartment(String) برای نوع Employee تعریف نشده است.
06:23	چون setDepartment method برای Employee class وجود ندارد.
06:30	پس خط ;("IT")emp2.setDepartment را حذف کنید.
06:37	برای پرینت کردن جزئیات System.out.println("Details of Manager class:" emp2.gtDetails()) semicolon را تایپ کنید.
06:55	برنامه را Save و Run کنید.
06:58	در اینجا در خروجی ما :Manager of را blank (خالی) بدست می آوریم.
07:04	چون department در Manager class را با استفاده از emp2 شروع نکرده ایم.
07:12	برای Demo اجازه دهید که department پیش‌فرض IT باشد.
07:17	پس به Manager class بروید و مقدار department را شروع کنید.
07:25	برنامه را Save و Run کنید.
07:28	ما خروجی Employee object که Employee class را اشاره می‌کند بدست می آوریم.
07:34	Manager object که به Employee class و Manager object که به Manager class اشاره می کنند.
07:42	در اینجا می‌بینیم که getDetails() method از manager class با emp2 خوانده می شود.
07:49	اما وقتی که emp2 می‌خواهد setDeparment را بخواند یک اشتباه می دهد.
07:54	که به این دلیل است: کامپایلر getDetails() method را در Employee class در زمان emp2.getDetails() می بیند.
08:05	پس این اشتباه نمی‌دهد و کد را معتبر می کند.
08:10	در زمان run پس JVM که getDetails() را در Manager class بعنوان getDetails() از Manager class که getDetails() از Employee class را override می کند.
08:24	پس ما خروجی را طبق getDetails() از Manager class بدست می آوریم. اما کامپایلر setDepartment method را در Employee class نمی بیند.
08:36	بنابراین باعث اشتباه در setDepartment که با emp2 خوانده می شود, خواهد شد.
08:43	اینجا Employee method getDetails() برای Employee class فراخوانی می شود.

08:49	مرجع کامپایلر که Employee class برای ()getDetails در زمان ()emp1.getDetails می باشد.
08:57	در زمان اجرا JVM که ()getDetails در Employee class را فراخوانی می‌کند. پس ما خروجی را طبق ()getDetails از Employee class بدست می آوریم.
09:08	پس JVM که متد مناسب برای object که در هر متغیر اشاره شده است را می خواند.
09:16	به این رفتار Virtual Method Invocation گفته می شود.
09:21	و به methods که Virtual Methods گفته می شود.
09:26	تمام متدها در java به این صورت می باشند.
09:31	ما یاد گرفتیم که Virtual Method Invocation چیست
09:36	و از قبل Compile-time polymorphism که method overloading است را یاد گرفتیم.
09:42	و به طور خلاصه Compile time polymorphism را یاد می گیریم.
09:47	در Compile time polymorphism که class می‌تواند بیشتر از یک method داشته باشد.
09:53	متد شامل همان نام اما با تعداد متفاوتی arguments می باشد.
09:59	کامپایلر قادر به تشخیص method call در compile-time می‌باشد و به این دلیل است که به آن compile time polymorphism گفته می شود.
10:09	خلاصه می کنیم
10:11	در این برنامه این موارد را یاد گرفتیم: polymorphism در Java چیست Run-time polymorphism Virtual Method Invocation و Compile-time polymorphism
10:23	ارائه: متدها را برای Vehicle و Bike class که در برنامه‌های قبلی استفاده شده را override کنید.
10:32	ویدئو در لینک زیر خلاصه Spoken Tutorial Project می باشد, لطفاً مشاهده کنید.
10:40	تیم این پروژه کارگاه آموزشی استفاده از Spoken Tutorial ارائه می دهد. و به کسانی که آزمون آنلاین را قبول شوند, گواهینامه می‌دهد . برای جزئیات بیشتر لطفاً به ما ایمیل بفرستید .

بودجه Spoken Tutorial Project توسط NMEICT, MHRD دولت هند تأمین می شود.اطلاعات بیشتر در لینک نشان داده شده در دسترس می باشد.	10:51
ترجمه و صداگذاری شبنم اقبال خان. با تشکر از شما	11:03