|  |  |
| --- | --- |
| **1. phân biệt Errors và Exception?** | + Error : Những lỗi nghiêm trọng không thể sử lý được,chương trình sẽ bị chết .  Ví dụ : Lỗi RAM , lỗi máy ảo JVM, tràn bộ nhớ StackOverFlow.  +Exxeption : Những bất thường xảy ra mà có thể xử lý được. |
| **2. Exception là gì ? Các loại Exception ?** | Exception : là những bất thường xảy ra và có thể sử lý được.  Các loại Exception :  + Checked: xảy ra tại thời điểm biên dịch (compiler-time), yêu cầu cần phải xử lý thì chương trình mới có thể chạy được  + Unchecked : xảy ra tại thời điểm thực thi chương trình , không cần phải sử lý chương trình vẫn chạy được. Lưu ý : khuyến cáo nên sử lý 🡺 nếu không sử lý , khi gặp Exception sẽ chết chương trình |
| **3. Các cách xử lý Exception ? khi nào xử dụng cái nào ?** | Có 2 cách sử lý :  Cách 1 : xử lý tại chỗ (try-catch)  🡺 Những đoạn code nghi ngờ xảy ra ngoại lệ thì bỏ vào khối try  🡺 Những công việc cần làm khi ngoại lệ xảy ra thì viết trong khối catch  Cách 2 : Sử dụng thows để ném ngoại lệ ra bên ngoài method để nơi nào sử dụng method đó sẽ xử lý |
| **4. Lưu ý khi sử dụng try-catch?** | + Trong một khối try có thể có nhiều khối catch nhưng chỉ có một khối finally  + Tại một thời điểm chỉ có một Exception xảy ra và một khối catch thực thi  + Các khối catch cần phải được sắp xếp từ exception con đến cha  + Khối finally luôn luôn chạy cho dù exception có xảy ra hay không. |
| **5. Try-with-resource dùng để làm gì ? cách sử dụng** | Try-with-resource **:** trong Java giúp tự động đóng tài nguyên sau khi sử dụng, giúp tránh rò rỉ và giảm mã dư thừa.  Cách sử dụng:  Cú pháp của try-with-resources sử dụng một khối try mà bên trong có khai báo tài nguyên (ví dụ: FileReader, BufferedReader, Connection, Socket...), và các tài nguyên này phải implement AutoCloseable. |
| **6. Khi nào sử dụng finally** | -Khi muốn đảm bảo một đoạn mã luôn được thực hiện (dù có lỗi hay không). -Thường dùng để giải phóng tài nguyên như đóng file, đóng kết nối CSDL, .. |