Bij de uitvoering van programmacode kan het gebeuren dat doorgaan volgens het normale executiepad niet mogelijk is. Als dit in een Java-programma gebeurt, dan treedt er een error of een exception op. Tussen deze begrippen zit een duidelijk verschil. Het gebruik van het woord 'exception' geeft iets meer speelruimte voor het benoemen van de situaties dan het woord 'error'.

Exception Handling: wat, wanneer en hoe

Fouten tijdig afvangen én melden

Jeroen Meetsma
is softwarearchitect
bij IJsberg ICT

Een error is in Java een uitzonderingssituatie die aangeeft dat de technische integriteit van de applicatie niet in orde is; een method die wordt aangeroepen blijkt niet te bestaan of bytecode is corrupt. Ook kan het zijn dat er problemen zijn met de systeem resources. De overige uitzonderingssituaties, met 'exception' aangeduid, kunnen in drie categorieën worden verdeeld:

1. Een exceptie als gevolg van een programmeer- of constructiefout.
2. Een exceptie als gevolg van een uitzonderlijke maar reguliere situatie.
3. Een exceptie als gevolg van het feit dat een (eigen) applicatieonderdeel dienst weigert.

'Exception handling' kent drie aspecten:

1. Het onderkennen en benoemen van uitzonderlijke situaties.
2. Het uittreden uit de normale programmaflow middels het laten optreden (raise of throw) van een exception of error.
3. Het afvangen van dergelijke situaties, gevolgd door een nette afhandeling.

Het doel van exception handling is het robuust maken van de applicatie. Dit is enerzijds het voorkomen van schade en anderzijds het rapporteren van problemen naar belanghebbenden. Er moet voorkomen worden dat er bijvoorbeeld corrupte data wordt gepersisteerd of dat de applicatie in een oneindige lus blijft hangen.

Daarnaast moet een storing in het programmaverloop worden gerapporteerd. Mogelijke belanghebbenden zijn gebruikers, beheerders of operators en een onderhoudsteam. Een gebruiker is gebaat bij een controlled response in de user interface. Een beheerder moet soms gealarmeerd worden als er een situatie optreedt die direct ingrijpen met beheertools vereist. Ontwikkelaars hebben baat bij accurate informatie in logfiles.

Het is aan de ontwikkelaar om te bedenken welke foutsituaties zich mogelijkerwijs kunnen voordoen en wat er in dergelijke gevallen moet gebeuren. Er zijn twee situaties denkbaar waarin een exceptie mag worden gegooid:

1. Code wordt aangeroepen op een manier die niet voldoet aan gestelde precondities.
2. De omgeving verkeert in een staat die het onmogelijk maakt de gevraagde taak uit te voeren. Het kan zowel een directe resource betreffen als een interne of externe API.

Design by contract

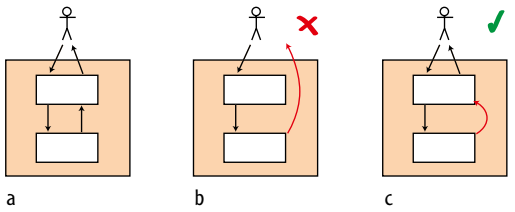
Een exceptie kan impliciet ontstaan of zelf worden geïnstantieerd. Een ontwikkelaar hoeft niet alle foutsituaties te adresseren om bruikbare code op te leveren. Zo levert het voortdurend controleren van alle inputparameters van een methode veel te veel zinloze code op. Als gewerkt wordt volgens het principe design by contract, is het immers aan de aanroepende partij om zich aan het contract te houden. Alleen als een parameter een reëel probleem vormt, is het zinvol om een IllegalArgumentException te gooien. Meestal is dit niet nodig: als een invoerparameter een verboden waarde als null bevat, levert dit dikwijls in de code een bruikbare NullPointerException op.

Voor het afhandelen van excepties geldt dat de applicatie zich naar buiten toe consistent en betrouwbaar moet gedragen. Verder moeten alle voorkomende situaties precies worden gelogd, zodat het mogelijk is code later aan te passen aan uitzonderingssituaties die zich in de praktijk voordoen. Een degelijk opgezet raamwerk voor foutaf-

handeling en naleving van enkele conventies is een absolute randvoorwaarde voor het maken van robuuste applicaties.

Conventies

Een applicatie is een black box voor gegevensinvoer en -uitvoer. Invoerhandelingen leveren doorgaans een response als uitvoer op. De applicatie is intern ook weer opgebouwd uit black boxes, ondergebracht in methodes. De aan een request verbonden invoer en uitvoer doorloopt binnen de applicatie een program flow waarvan de doorlopen methodes een call stack vormen (zie figuur a). Een exceptie is een response die staat voor een abnormale beëindiging van de program flow en die in staat is meerdere niveaus in de call stack terug te springen.

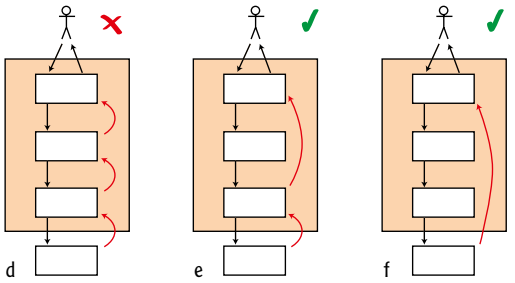


Bij de opzet van conventies voor exception handling moet rekening worden gehouden met enkele principes:

- Een eindgebruiker, of een ander aanroepend systeem, moet altijd een gecontroleerde response terugkrijgen. Een response die past in de gehanteerde interface (zie figuur c).
- Een exceptie mag nooit worden genegeerd; oorsprong en omstandigheden ervan moet altijd terug te vinden zijn in een logbestand.
- Applicatieonderdelen moeten geen details, zoals specifieke excepties, kennen van diepere onderdelen in een call stack, omdat implementatiedetails in de loop der tijd kunnen veranderen.
- Verder is het in het algemeen raadzaam om code zo simpel en compact mogelijk te houden.

Deze principes leiden tot de volgende conventies:

- Een exceptie zelf is niet bruikbaar als response naar een eindgebruiker (zie figuur b). Één niveau lager moet een exceptie worden afgevangen en omgezet naar een bericht op de (user) interface.



- Indien de applicatie een gebruiker de mogelijkheid biedt de exceptie te melden aan een systeembeheerder, is het goed deze te voorzien van een uniek ID.
- Dit vangnet moet ook worden gebruikt om alle excepties (en geneste oorzaken) te loggen die dieper in de callstack nog niet waren afgevangen.
- Een (uit een API afkomstige) (checked) exceptie die wordt afgevangen moet altijd ofwel worden gelogd en afgehandeld ofwel in één of andere vorm worden doorgegooid.
- Het afhandelen van een exceptie betekent dat de uitzonderlijke omstandigheden worden weggenomen of vermeden en dat de program flow zijn normale weg kan vervolgen.
- Runtime excepties en errors mogen niet anders dan door het bovengenoemde vangnet worden afgevangen tenzij zij specifiek kunnen worden afgevangen en er een afhandeling mogelijk is. Damage control moet worden gegarandeerd middels code in finally blocks.
- Impliciete afvanging van runtime excepties en errors door 'catch (Exception)' of 'catch (Throwable)' is, met uitzondering van het vangnet, uit den boze.
- Het doorgooien van een exceptie geschiedt bij voorkeur door deze te verpakken in een eigen applicatiespecifieke runtime exception (zie figuur e). Dit heeft drie redenen:
  1. De bovenliggende elementen in de call stack hoeven geen kennis te hebben van implementatiespecifieke excepties.
  2. Het is ook niet nodig om als alternatief een eigen exceptiehiërarchie te ontwikkelen voor elk niveau in de call stack. Dit zou een enorme hoeveelheid extra catch blocks opleveren (zie figuur d).
  3. Het hierboven genoemde vangnet draagt zorg voor een nette afhandeling.
- Ook eigen excepties worden bij voorkeur als runtime exceptie gegooid, tenzij de code die de exceptie declareert, bedoeld is als API.
- Runtime excepties afkomstig uit een API hoeven niet direct te worden afgevangen (figuur f), tenzij gerichte afhandeling mogelijk is.

Verder geldt dat exception handling niet wordt gebruikt om applicatielogica of gebruikersfunctionaliteit te implementeren. Dit geldt bijvoorbeeld voor applicatieve invoer. Voor alle invoer intern in de applicatie geldt dat bij het niet voldoen aan precondities een runtime exceptie mag worden gegooid.

Invoer van buitenaf, bijvoorbeeld een browser van een internetgebruiker, varieert in betrouwbaarheid van 'in principe compliant' tot 'potentieel corrupt'. Exception handling is hiervoor niet bedoeld. Invoer van buitenaf moet met filtering, validatie, en conversie geschikt worden gemaakt voor verwerking. «

Voor invoer in de applicatie mag een runtime exceptie worden gegooid.